A characterization of symmetric semantics by domain complementation

Roberto Giacobazzi
Dipartimento Scientifico e Tecnologico
Università di Verona
Strada Le Grazie, I-37134 Verona, Italy
giaco@sci.univr.it

Isabella Mastroeni
Dipartimento Scientifico e Tecnologico
Università di Verona
Strada Le Grazie, I-37134 Verona, Italy
mastroen@arena.sci.univr.it

ABSTRACT

We characterize the symmetric structure of Cousot's hierarchy of semantics in terms of a purely algebraic manipulation of abstract domains. We consider domain complementation in abstract interpretation as a formal method for systematically deriving complementary semantics of programming languages. We prove that under suitable hypothesis the semantics abstraction commutes with respect to domain complementation. This result allows us to prove that angelic and demonic/infinite semantics are complementary and provide a minimal decomposition of all natural-style trace-based, relational, denotational, Dijkstra's predicate transformer and Hoare's axiomatic semantics. We apply this construction to the case of concurrent constraint programming, characterizing well known semantics as abstract interpretation of maximal traces of constraints.

Categories and Subject Descriptors

D.3 [**Programming languages**]: Formal definitions and theory—*Semantics*; F.3 [**Logics and meanings of programs**]: Semantics of Programming Languages—*algebraic approaches to semantics, denotational semantics, operational semantics*

General Terms

Theory

Keywords

Abstract interpretation, domain complementation, program semantics, concurrent constraint programming.

1. INTRODUCTION

Since its origin in 1977, abstract interpretation [9] has been widely used, implicitly or explicitly, to describe and formalize approximate computations in many different areas of computer science, from its very beginning use in formalizing (compile-time) program analysis frameworks to more recent applications in model checking, program verification, data security, type inference, automated deduction, and comparative semantics. This justifies a now well established definition of abstract interpretation as *a general theory to approximate the semantics of discrete dynamic systems"* [7].

Semantics is central in the construction of an abstract interpretation. The so called *concrete semantics* specifies the observable property of program behavior and any more abstract semantics, e.g. decidable semantics for program analysis, can be derived by abstraction. As a consequence, semantics, at any level of abstraction, can be fully specified as an abstract interpretation of a more concrete base semantics. This key idea is the basis of Cousot's design of a complete hierarchy of semantics of programming languages [12, 8]. A number of semantics including big-step, termination and nontermination, Plotkin's natural, Smyth's demonic, Hoare's angelic relational and corresponding denotational, Dijkstra's predicate transformer weakest-precondition and weakest-liberal precondition and Hoare's partial and total axiomatic semantics, have all been derived by successive abstractions from an (operational) maximal trace semantics of a transition system. The resulting hierarchy (here called Cousot's hierarchy) provides a complete account on the structure and the relative precision of most well known semantics of programming languages. One of the major challenges of this construction is that semantics can be fully characterized as domain abstractions, and therefore they can be manipulated in a purely algebraic way by modifying domains. This view of semantics as domains provides both a better insight on the structure and relative precision of traditional well known semantics of programming languages and the possibility to systematically specify new semantics by composition, decomposition, refinement and simplification of existing ones, by manipulating the corresponding domains.

1.1 The main results

In this paper we analyze Cousot's hierarchy of semantics and we characterize its symmetric structure (see Fig. 1) in terms of a purely algebraic manipulation of domains. We consider *abstract domain complementation* introduced in [6] as the basic formal method for decomposing semantics and *reduced product* of domains introduced in [11] as the basic operation for composing semantics. Given any two semantics $\mathbb{S}_1$ and $\mathbb{S}_2$, the product semantics $\mathbb{S}_1 \sqcap \mathbb{S}_2$ is the

PPDP '00, Montréal, Canada.

most abstract semantics which is as precise as both $\mathcal{S}_1$ and $\mathcal{S}_2$, namely which is able to observe both the observables of $\mathcal{S}_1$ and $\mathcal{S}_2$. Domain complementation was originally introduced to decompose abstract domains in static program analysis, and it is the inverse operation to reduced product. In our case, this operation provides a systematic methodology for decomposing semantics by characterizing the most abstract semantics $\mathcal{S}$ which, when composed with a given semantics $\mathcal{B}$, yields the semantics $\mathcal{C} = \mathcal{S} \sqcap \mathcal{B}$ as result. These operations provide advanced methods for comparing semantics with respect to their relative expressiveness. This kind of information is particularly relevant when studying hierarchies of semantics, in particular in studying semantics observing complementary behaviors of programs, e.g. finite and infinite computations of a transition system. According to Cousot's construction in fact, any semantic style (trace-operational, relational, denotational, Dijkstra's predicate transformer and Hoare's axiomatic semantics) may have a corresponding *natural*, *finite/angelic*, *demonic*, and *infinite* nature. The nature of each semantics defines a corresponding observable behavior of programs (later called *observable*), which can be parameterized according to the chosen semantic style, and it corresponds respectively to: finite and infinite, terminating, chaotic nonterminating, and infinite computations. We prove that this highly symmetric structure is a consequence of a common pattern of abstraction between semantic styles and observables, ranging from operational trace-based to the more abstract Hoare's axiomatic semantics. We characterize this pattern in terms of some basic properties of the closure operators associated with the semantics abstractions in Cousot's hierarchy. This allows us to prove the basic results on symmetric semantics for the trace-operational semantic style only, deriving the results concerning all the other styles and observables as a simple consequence. In particular we prove that natural semantics are always the reduced product of finite/angelic and demonic or infinite semantics, and that these semantics factorize the natural semantic construction by complementation. We prove that any finite/angelic semantics can be systematically derived as the domain complementation in the natural semantics of the demonic or infinite semantics. Moreover, we prove that finite/angelic and infinite semantics always form a most abstract decomposition of any natural semantics, and that demonic semantics can be further factorized in terms of infinite semantics and a new semantics, here called *pseudo-demonic*, which is unable to observe infinite computations in presence of finite chaos, namely when programs may produce any possible output in finite time. These results provide both an algebraic characterization of complementary observable properties in semantics and a decomposition result for observable properties of programs in terms of complementary observables, similarly to the well-known Alpern & Schneider's safety/liveness decomposition of properties of concurrent program executions (cf. [2]). Our results are general and can be applied to any programming languages whose semantics can be defined as traces of program states in a transition system.

We apply this construction to the case of concurrent constraint programming *ccp* languages [28]. These languages well fit into Cousot's hierarchy of semantics being easily defined in terms of traces of constraints in a transition system. We prove that both Saraswat's closure-based denotational [29] and de Boer et al.'s predicate transformer semantics [13] can be derived by abstract interpretation from maximal traces of constraints associated with an operational small-step transition system semantics of *ccp* programs, and consequently the corresponding finite/angelic, demonic, and infinite semantics can be specified by domain complementation.

The paper is organized as follows. In Sec. 2 we recall the basic theory of abstract interpretation. Cousot's hierarchy of semantics is summarized in Sec. 3, while in Sec. 4 we recall the basic notion of domain complementation. In Sec. 5 we prove some basic results concerning closure operators and domain complementation. These results are the basis for generalizing the results in Sec. 6 and characterizing the symmetric structure of different observables, to the whole hierarchy of semantics. This generalization is in Sec. 6.2. In Sec. 6.3, we consider the complementation of semantics with respect to a different order, which is used as computational order in Cousot's hierarchy, and in Sec. 7 we apply our results to the case of concurrent constraint programming. Sec. 8 concludes.

1.2 State of the art

The foundation for a theory of abstract domains was fixed by Cousot & Cousot in [11]. The authors gave the main structure of abstract domains enjoying *Galois connections* and some basic operators to systematically compose domains, respectively the *reduced product* and *reduced power* operations. Since then, a number of papers have developed new domain operations and studied the impact of these operations in the design of abstract interpretations (e.g. see [21] for a survey). The notion of *domain refinement* and *domain simplification*, introduced in [16, 21], provided the very first generalization of these ideas. Intuitively a refinement is any operator performing an action of refinement with respect to the standard order of precision, e.g. by adding information to domains; while simplificators perform the dual action of "taking out" information from domains. Few examples are known on the use of systematic domain operations in abstract interpretation to reason about the structure and the expressiveness of semantics of programming languages. Most of these examples are in the semantics of logic programs, which basically relies on the hierarchy of semantics developed in [5, 18]. In [20] the authors study the relations between different semantics of logic programs, namely success pattern semantics, computed answer substitution semantics and call pattern semantics by means of complementation. This is the very first and unique example of the use of complementation in systematic semantics design. In [22], the authors proved that compositional (denotational) semantics of logic programs can be systematically designed by a generalization of Cousot's reduced cardinal power operation [11] from non compositional semantics of computed answer substitution. To the best of our knowledge, the only use of domain operations in more general semantics constructions is in [8], where the domain operation of *tensor product* [30] is considered in order to design Hoare's axiomatic semantics.

2. PRELIMINARIES

2.1 Basic notions

If S and T are sets, then $\wp(S)$ denotes the powerset of S, $S \smallsetminus T$ denotes the set-difference between S and T, $S \subset T$ denotes strict inclusion, and for a function $f : S \rightarrow T$ and

$X \subseteq S$, $f(X) \stackrel{\text{def}}{=} \{f(x) \mid x \in X\}$. By $g \circ f$ we denote the composition of the functions f and g, i.e., $g \circ f \stackrel{\text{def}}{=} \lambda x.g(f(x))$. The notation $\langle P, \leq \rangle$ denotes a poset P with ordering relation $\leq$, while $\langle P, \leq, \vee, \wedge, \top, \bot \rangle$ denotes a complete lattice P, with ordering $\leq$, *lub* $\vee$, *glb* $\wedge$, greatest element (top) $\top$, and least element (bottom) $\bot$. Often, $\leq_P$ will be used to denote the underlying ordering of a poset P, and $\vee_P$, $\wedge_P$, $\top_P$ and $\bot_P$ denote the basic operations and elements of a complete lattice. The notation $C \cong A$ denotes that C and A are isomorphic ordered structures. $x \in C$ is *meet-irreducible* if $x = a \wedge b \;\Rightarrow\; x \in \{a, b\}$. The set of meet-irreducible elements in C is denoted $Mirr(C)$. The downward closure of S is defined as $\downarrow S \stackrel{\text{def}}{=} \{x \in P \mid \exists y \in S.\ x \leq_P y\}$, and for $x \in P$, $\downarrow x$ is a shorthand for $\downarrow \{x\}$, while the upward closure $\uparrow$ is dually defined. $S \longrightarrow T$ denotes the set of all functions from S to T. We use the symbol $\sqsubseteq$ to denote pointwise ordering between functions: If S is any set, P a poset, and $f, g : S \to P$ then $f \sqsubseteq g$ if for all $x \in S$, $f(x) \leq_P g(x)$. Let C and A be complete lattices. Then, $C \xrightarrow{\text{m}} A$, $C \xrightarrow{\text{c}} A$, $C \xrightarrow{\text{a}} A$, and $C \xrightarrow{\text{coa}} A$ denote, respectively, the set of all monotone, (Scott-)continuous, additive, and co-additive functions from C to A. Recall [1] that $f \in C \xrightarrow{\text{c}} A$ iff f preserves *lub*'s of (nonempty) chains iff f preserves *lub*'s of directed subsets, and $f : C \to A$ is (completely) additive if f preserves *lub*'s of all subsets of C (emptyset included). Co-additivity is defined by duality. We denote by $\mathit{lfp}(f)$ and $\mathit{gfp}(f)$, respectively, the least and greatest fixpoint, when they exist, of an operator f on a poset. If $f \in C \xrightarrow{\text{c}} C$ then $\mathit{lfp}(f) = \vee_{i \in \mathbb{N}} f^i(\bot_C)$, where, for any $i \in \mathbb{N}$ and $x \in C$, the i-th power of f in x is inductively defined as follows: $f^0(x) = x$; $f^{i+1}(x) = f(f^i(x))$. Dually, if f is co-continuous then $\mathit{gfp}(f) = \wedge_{i \in \mathbb{N}} f^i(\top_C)$. $\{f^i(\bot_C)\}_{i \in \mathbb{N}}$ and $\{f^i(\top_C)\}_{i \in \mathbb{N}}$ are called, respectively, the *upper* and *lower Kleene's iteration sequences of* f (see [10]).

2.2 Abstract interpretation

Abstract domains can be equivalently formulated either in terms of Galois connections or closure operators [11]. An *upper closure operator* on a poset P is an operator $\rho : P \to P$ monotone, idempotent and extensive ($\forall x \in P.\ x \leq_P \rho(x)$). *Lower closure operators* are defined dually. The set of all upper (lower) closure operators on P is denoted by $uco(P)$ ($lco(C)$). Let $\langle C, \leq, \vee, \wedge, \top, \bot \rangle$ be a complete lattice. A basic property of closure operators is that each closure is uniquely determined by the set of its fixpoints $\rho(C)$. For upper closures: $X \subseteq C$ is the set of fixpoints of an upper closure on C iff X is a *Moore-family* of C, i.e., $X = \mathcal{M}(X) \stackrel{\text{def}}{=} \{\wedge S \mid S \subseteq X\}$ — where $\wedge \varnothing = \top \in \mathcal{M}(X)$. For any $X \subseteq C$, $\mathcal{M}(X)$ is called the *Moore-closure* of X in C, i.e., $\mathcal{M}(X)$ is the least (w.r.t. set-inclusion) subset of C which contains X and it is a Moore-family of C. It turns out that $\langle \rho(C), \leq \rangle$ is a complete meet subsemilattice of C (i.e., $\wedge$ is its *glb*). Often, we will find particularly convenient to identify closure operators with their sets of fixpoints, possibly using as notation capital Latin letters. If C is a complete lattice then $uco(C)$ ordered pointwise is also a complete lattice, denoted by $\langle uco(C), \sqsubseteq, \sqcup, \sqcap, \lambda x.\top, \lambda x.x \rangle$, where for every $\rho, \eta \in uco(C)$, $\{\rho_i\}_{i \in I} \subseteq uco(C)$ and $x \in C$:

- $\rho \sqsubseteq \eta$ iff $\forall y \in C.\ \rho(y) \leq \eta(y)$ iff $\eta(C) \subseteq \rho(C)$;
- $(\sqcap_{i \in I} \rho_i)(x) = \wedge_{i \in I} \rho_i(x)$;
- $(\sqcup_{i \in I} \rho_i)(x) = x \;\Leftrightarrow\; \forall i \in I.\ \rho_i(x) = x$;

If $\alpha : C \xrightarrow{\text{m}} A$ and $\gamma : A \xrightarrow{\text{m}} C$ are monotone functions such that $\lambda x.x \sqsubseteq \gamma \circ \alpha$ and $\alpha \circ \gamma \sqsubseteq \lambda x.x$, then (A, α, γ, C) is called a *Galois connection* (GC for short) or *adjunction* between C and A, also denoted $\langle C, \leq_C \rangle \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \langle A, \leq_A \rangle$. Note that in GC, for any $x \in C$ and $y \in A$: $\alpha(x) \leq_A y \Leftrightarrow x \leq_C \gamma(y)$ and $\gamma(y) = \bigvee \{\, x \mid \alpha(x) \leq y \,\}$ and $\alpha(x) = \bigwedge \{\, y \mid x \leq \gamma(y) \,\}$. The set of all GC's between two complete lattices A and C is the tensor product $A \otimes C$, which is a complete lattice and $A \otimes C \cong A \xrightarrow{\text{a}} C \cong C \xrightarrow{\text{coa}} A$ [30]. If in addition $\alpha \circ \gamma = \lambda x.x$ ($\gamma \circ \alpha = \lambda x.x$), then (A, α, γ, C) is a *Galois insertion* (GI) (resp. *projection*) also denoted $\langle C, \leq_C \rangle \underset{\alpha}{\overset{\gamma}{\leftarrow\!\!\twoheadrightarrow}} \langle A, \leq_A \rangle$ (resp. $\langle C, \leq_C \rangle \underset{\alpha}{\overset{\gamma}{\twoheadleftarrow\!\!\rightarrow}} \langle A, \leq_A \rangle$) of A in C. It is worth noting that $A \cong C$ if and only if $\langle C, \leq_C \rangle \underset{\alpha}{\overset{\gamma}{\twoheadleftarrow\!\!\twoheadrightarrow}} \langle A, \leq_A \rangle$.
The standard abstract interpretation framework is based on the adjoint relation between abstraction and concretization functions [9]. The concrete and abstract domains, C and A, are assumed to be complete lattices and are related by abstraction and concretization maps forming a GC (A, α, γ, C). Following a standard terminology, A is called an abstraction of C, and C is a concretization of A. If (A, α, γ, C) is a GI, then each value of the abstract domain A is useful in representing C, because all the elements of A represent distinct members of C, being γ 1-1. Any GC may be lifted to a GI by identifying in an equivalence class those values of the abstract domain with the same concretization. This process is known as *reduction* of the abstract domain. Note that any GI (A, α, γ, C) uniquely determines an upper closure operator $\gamma \circ \alpha \in uco(C)$ and conversely, any closure operator $\rho \in uco(C)$ uniquely determines a GI $(\rho(C), \rho, id, C)$, up to isomorphic representation of domain's objects. Hence, we will identify $uco(C)$ with the so-called *lattice* $\mathfrak{L}_C$ *of abstract interpretations* of C (cf. [9, Section 7] and [11, Section 8]), i.e., the complete lattice of all possible abstract domains (modulo isomorphic representation of their objects) of the concrete domain C. The pointwise ordering on $uco(C)$ corresponds precisely to the standard ordering used to compare abstract domains with regard to their precision: A_1 is more precise than A_2 (i.e., A_2 is an abstraction of A_1) iff $A_1 \sqsubseteq A_2$ in $uco(C)$ iff $\langle A_1, \leq_{A_1} \rangle \underset{\alpha}{\overset{\gamma}{\leftarrow\!\!\twoheadrightarrow}} \langle A_2, \leq_{A_2} \rangle$. Let $\{A_i\}_{i \in I} \subseteq uco(C)$: $\sqcup_{i \in I} A_i$ is the most concrete among the domains in $\mathfrak{L}_C$ which are abstractions of all the A_i's, i.e., $\sqcup_{i \in I} A_i$ is the *least* (w.r.t. $\sqsubseteq$) *common abstraction* of all the A_i's; and $\sqcap_{i \in I} A_i$ is (isomorphic to) the well-known *reduced product* (basically cartesian product plus reduction) of all the A_i's, or, equivalently, it is the most abstract among the domains in $\mathfrak{L}_C$ which are more concrete than every A_i. Let us remark that the reduced product can be also characterized as Moore-closure of set-union, i.e., $\sqcap_{i \in I} A_i = \mathcal{M}(\cup_{i \in I} A_i)$.

3. COUSOT'S SEMANTICS HIERARCHY

In this section we recall Cousot's hierarchy of semantics [12, 8]. We consider the operational semantics of a programming language that associates a discrete transition system with each well-formed program. In other words we associate a pair $\langle \Sigma, \tau \rangle$ where Σ is a nonempty set of states and $\tau \subseteq \Sigma \times \Sigma$ is a binary transition relation between a state and its possible successors. In the following, $\Sigma^{\vec{+}}$ and $\Sigma^{\vec{\omega}} \stackrel{\text{def}}{=} \mathbb{N} \longrightarrow \Sigma$ denote respectively the set of finite nonempty and infinite sequences of symbols in Σ. The length of a sequence $\sigma \in \Sigma^{\vec{\infty}} \stackrel{\text{def}}{=} \Sigma^{\vec{+}} \cup \Sigma^{\vec{\omega}}$ is denoted $|\sigma| \in \mathbb{N} \cup \{\omega\}$ and its i-th element is denoted σ_i. A non-empty finite (infinite) *trace* σ is a finite (infinite) sequence of program states where two consecutive elements are in the transition relation τ, i.e., for all $i < |\sigma|$: $\langle \sigma_i, \sigma_{i+1}\rangle \in \tau$. A *generic trace* is any such element in $\Sigma^{\vec{\infty}}$. The *maximal trace semantics* of a transitions system [12] is $\tau^{\vec{\infty}} \stackrel{\text{def}}{=} \tau^{\vec{+}} \cup \tau^{\vec{\omega}}$, where if $T \subseteq \Sigma$ is a set of final/blocking states $\tau^{\vec{n}} = \{\sigma \in \Sigma^{\vec{+}} \mid |\sigma| = n, \forall i \in [0, n) \,.\, \langle \sigma_i, \sigma_{i+1}\rangle \in \tau\}$, $\tau^{\vec{\omega}} = \{\sigma \in \Sigma^{\vec{\omega}} \mid \forall i \in \mathbb{N} \,.\, \langle \sigma_i, \sigma_{i+1}\rangle \in \tau\}$, and $\tau^{\vec{+}} = \cup_{n>0}\{\sigma \in \tau^{\vec{n}} \mid \sigma_{n-1} \in T\}$. $\tau^{\vec{\infty}}$ has been obtained in [12] as a fixpoint of a monotone operator $F^{\vec{\infty}} : \wp(\Sigma^{\vec{\infty}}) \to \wp(\Sigma^{\vec{\infty}})$ which is defined by a biinductive (inductive and co-inductive) definition on $\langle \wp(\Sigma^{\vec{\infty}}), \sqsubseteq^{\vec{\infty}}, \sqcap^{\vec{\infty}}, \sqcup^{\vec{\infty}}, \sqcap^{\vec{\infty}}, \Sigma^{\vec{+}}, \Sigma^{\vec{\omega}}\rangle$, where $X \sqsubseteq^{\vec{\infty}} Y$ iff $X \cap \Sigma^{\vec{+}} \subseteq Y \cap \Sigma^{\vec{+}}$ and $Y \cap \Sigma^{\vec{\omega}} \subseteq X \cap \Sigma^{\vec{\omega}}$. This allows us to combine both least and greatest fixpoint in a unique fixpoint presentation: finite (terminating) traces are obtained by induction (*lfp*) of $F^{\vec{\infty}}$ on $\langle \wp(\Sigma^{\vec{+}}), \subseteq\rangle$ and infinite traces are obtained by co-induction (*gfp*) on $\langle \wp(\Sigma^{\vec{\omega}}), \subseteq\rangle$, which corresponds to the *lfp* of $F^{\vec{\infty}}$ in $\langle \wp(\Sigma^{\vec{\omega}}), \supseteq\rangle$. In this case: $\tau^{\vec{\infty}} = \mathit{lfp}(F^{\vec{\infty}})$ (see [12] for details).

All semantics in the hierarchy are derived as abstract interpretation of the trace-based semantics. Cousot distinguishes between an approximation order, which is set inclusion on sets of generic traces, and a computational order which corresponds to the lifted order $\sqsubseteq^{\vec{\infty}}$ characterizing finite and infinite traces as fixpoint of $F^{\vec{\infty}}$. The basic semantics in Cousot's hierarchy are summarized in Table 1. Each semantics in natural style corresponds to a suitable abstraction of the basic natural trace-based semantics $\tau^{\vec{\infty}}$. The *relational semantics* $\mathcal{R}$ associates an input-output relation with program traces by using the bottom symbol $\perp \notin \Sigma$, to denote nontermination. This corresponds to an abstraction of the maximal trace semantics where intermediate computation states are ignored. The *denotational semantics* $\mathcal{D}$ abstracts away from the history of computations by considering input-output functions. This semantics is isomorphic to relational semantics. Dijkstra's predicate transformer *gwp* and Hoare's axiomatic semantics $g\mathcal{H}$ are represented respectively as co-additive functions, denoting weakest-precondition predicate transformers, and elements in tensor product domains, i.e., GC's, specifying the adjoint relation between weakest-precondition and strongest-postcondition in Hoare's triples $\{P\}\ C\ \{Q\}$ (see [8]).

Each semantics in natural style may have a corresponding *angelic*, *demonic*, or *infinite* observable which is again an abstraction. We recall here only the corresponding abstractions for the trace-based semantics.

Angelic. The angelic trace semantics $\tau^{\vec{+}}$ is designed as an abstraction of the maximal trace semantics, and it is obtained by approximating sets of possibly infinite traces with their finite ones, i.e.,

$$\langle \wp(\Sigma^{\vec{\infty}}), \subseteq\rangle \xleftarrow[\alpha^{+}]{\gamma^{+}} \langle \wp(\Sigma^{\vec{+}}), \subseteq\rangle \quad \text{where} \quad \alpha^{+}(X) = X \cap \Sigma^{\vec{+}}$$

We denote by $\mathcal{R}^{+}$, $\mathcal{D}^{+}$, *wlp*, and $p\mathcal{H}$ respectively the big-step relational semantics [27], angelic denotational, weakest-liberal precondition [14], and Hoare's partial correctness semantics [25]. All these semantics have been proved by Cousot [8] to be the angelic abstractions of the corresponding semantics in natural style.

Demonic. The demonic trace semantics, denoted as $\tau^{\vec{\partial}}$, is derived from the maximal trace semantics by approximating the nontermination by chaos $\langle \wp(\Sigma^{\vec{\infty}}), \subseteq\rangle \xleftarrow[\alpha^{\partial}]{\gamma^{\partial}} \langle D^{\vec{\partial}}, \subseteq\rangle$, where $D^{\vec{\partial}} \subseteq \wp(\Sigma^{\vec{\infty}})$ is such that $X \in D^{\vec{\partial}}$ iff

$$\sigma \in X \ \wedge\ |\sigma| = \omega \ \Rightarrow\ \forall \delta \in \Sigma^{\vec{+}} .\, \sigma_0\delta \in X$$

and $\alpha^{\partial}(X) = X \cup \left\{ \sigma_0\delta \,\middle|\, \sigma \in X, |\sigma| = \omega, \delta \in \Sigma^{\vec{+}} \right\}$. We denote by $\mathcal{R}^{\partial}$, $\mathcal{D}^{\partial}$, *wp*, and $t\mathcal{H}$ the demonic relational, demonic denotational [3], weakest-precondition [14] and Hoare's total correctness semantics [25]. These semantics have been proved by Cousot [8] to be the demonic abstractions of the corresponding semantics in natural style.

Infinte. The infinite trace semantics, denoted $\tau^{\vec{\omega}}$, is derived by observing nonterminating traces only, namely by the following abstraction $\langle \wp(\Sigma^{\vec{\infty}}), \subseteq\rangle \xleftarrow[\alpha^{\omega}]{\gamma^{\omega}} \langle \wp(\Sigma^{\vec{\omega}}), \subseteq\rangle$ where $\alpha^{\omega}(X) = X \cap \Sigma^{\vec{\omega}}$. The corresponding infinite semantics are denoted by $\mathcal{R}^{\omega}$, $\mathcal{D}^{\omega}$, wp^{ω}, and $t\mathcal{H}^{\omega}$.

The whole hierarchy, relating semantics styles and observables is shown in Fig. 1, where continuous lines and arrows denote, respectively, isomorphisms and strict abstractions between semantics.

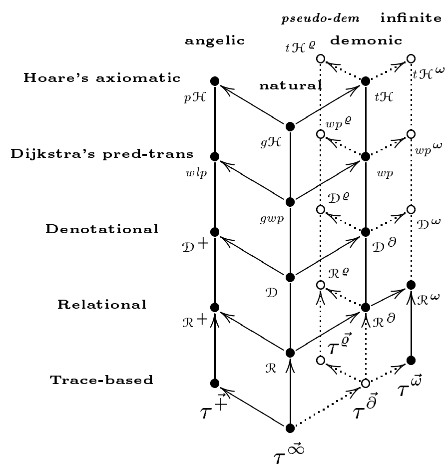

Figure 1: Semantics in Cousot's hierarchy.

4. DOMAIN COMPLEMENTATION

A sequence of abstract domains $\{A_i\}_{i\in I}$ is a (conjunctive) decomposition of the abstract domain B if $B = \sqcap_{i\in I} A_i$. Abstract domain complementation introduced in [6] provides a systematic method for decomposing abstract domains. Complementation is the *inverse* operation of the reduced product (see [21]), namely an operation which, starting from any two domains $C \sqsubseteq D$, gives as result the most abstract domain $C \ominus D$, whose reduced product with D is exactly C (i.e., $(C \ominus D) \sqcap D = C$). By the equivalence between closure operators and abstract domains, the above notion of complementation corresponds precisely to *pseudo-complementation* for the closure ρ_D corresponding to D in $uco(C)$. Recall that if L is a meet-semilattice with bottom then the *pseudo-complement* of $x \in L$, if it exists, is the

Semantics	*Domain relation*	*Abstraction*
$\mathcal{R} = \alpha^r(\tau^{\vec{\infty}})$	$\langle \wp(\Sigma^{\vec{\infty}}), \subseteq \rangle \underset{\alpha^r}{\overset{\gamma^r}{\leftrightarrows}} \langle \wp(\Sigma \times \Sigma_\bot), \subseteq \rangle$	$\alpha^r(X) = \{\ \langle \sigma_0, \sigma_{n-1} \rangle \mid\ \sigma \in X, \lvert\sigma\rvert = n\ \} \cup$ $\{\ \langle \sigma_0, \bot \rangle \mid\ \sigma \in X, \lvert\sigma\rvert = \omega\ \}$
$\mathcal{D} = \alpha^{\blacktriangleright}(\mathcal{R})$	$\langle \wp(\Sigma \times \Sigma_\bot), \subseteq \rangle \underset{\alpha^{\blacktriangleright}}{\overset{\gamma^{\blacktriangleright}}{\leftrightarrows}} \langle \Sigma \longrightarrow \wp(\Sigma_\bot), \sqsubseteq \rangle$	$\alpha^{\blacktriangleright}(X) \stackrel{\text{def}}{=} \lambda s. \{ s' \in \Sigma \mid \langle s, s' \rangle \in X \}$
$gwp = \alpha^{gwp}(\mathcal{D})$	$\langle \Sigma \longrightarrow \wp(\Sigma_\bot), \sqsubseteq \rangle \underset{\alpha^{gwp}}{\overset{\gamma^{gwp}}{\leftrightarrows}} \langle \wp(\Sigma_\bot) \xrightarrow{\text{coa}} \wp(\Sigma), \sqsupseteq \rangle$	$\alpha^{gwp}(f) = \lambda P.\ \{\ s \mid\ f(s) \subseteq P\ \}$
$g\mathcal{H} = \alpha^{gH}(gwp)$	$\langle \wp(\Sigma_\bot) \xrightarrow{\text{coa}} \wp(\Sigma), \sqsupseteq \rangle \underset{\alpha^{gH}}{\overset{\gamma^{gH}}{\leftrightarrows}} \langle \wp(\Sigma) \otimes \wp(\Sigma_\bot), \supseteq \rangle$	$\alpha^{gH}(\Phi) = \{\ \langle X, Y \rangle \in \wp(\Sigma) \times \wp(\Sigma_\bot) \mid\ X \subseteq \Phi(Y)\ \}$

Table 1: Basic natural-style semantics as abstract interpretations

unique element $x^* \in L$ such that $x \wedge x^* = \bot$ and $\forall y \in L.\ (x \wedge y = \bot) \Rightarrow (y \leq x^*)$ [4]. In a complete lattice L, if x^* exists then $x^* = \vee\{y \in L \mid x \wedge y = \bot\}$. If every $x \in L$ has the pseudo-complement, L is *pseudo-complemented*. It is worth noting that pseudo-complementation is the only possible form of complementation for abstract interpretation. Indeed, it is well known [15, 26] that $uco(C)$ is complemented (in the standard sense) iff C is a complete well-ordered chain, and this is a far too restrictive hypothesis for semantic domains. The following results [19, 17] provide sufficient conditions on C such that $uco(C)$ is pseudo-complemented. Recall that a complete lattice C is *meet-continuous* if for any chain $Y \subseteq C$ and $x \in C$, $x \wedge (\vee Y) = \vee_{y \in Y}(x \wedge y)$. Moreover C is meet-generated by $S \subseteq C$ if $C = \mathcal{M}(S)$.

THEOREM 4.1. *Let C be a complete lattice.*

1. *If C is a meet-continuous then $uco(C)$ is pseudo-complemented [19].*
2. *If C is meet-generated by $Mirr(C)$ then $uco(C)$ is pseudo-complemented and, for any $A \in uco(C)$, we have $A^* \stackrel{def}{=} C \ominus A = \mathcal{M}(Mirr(C) \smallsetminus A)$ [17].*

Note that for any $A, B \in uco(C)$ such that $A \sqsubseteq B$, $A \ominus B$ is well defined if $\uparrow A = uco(A)$ is pseudo-complemented.

5. SYMMETRIC ABSTRACTIONS

In the following we characterize the symmetric structure of Cousot's hierarchy of semantics in terms of a general algebraic property of closure operators. The fact that the same angelic/demonic/infinite observables are abstractions of the natural semantics in any style (trace-based, relational, denotational, predicate transformer and axiomatic) is indeed a consequence of the fact that the abstractions that relates the different styles of semantics and observables commute all over Cousot's hierarchy.

In order to better understand this situation we suppose to have a concrete semantic domain C which is generated by its meet-irreducibles, and two closure operators $\rho, \eta \in uco(C)$. Recall that $\rho \circ \eta \in uco(C)$ iff $\rho \circ \eta = \eta \circ \rho = \eta \sqcup \rho$ [26]. Therefore the structure in Fig. 2 is perfectly symmetric if the closure $\rho(C) \ominus \rho(\eta(C))$ corresponds precisely to the abstraction by ρ of the complementary closure $C \ominus \eta(C)$, namely if ρ is an $\ominus$-morphism: $\rho(C \ominus \eta(C)) = \rho(C) \ominus \rho(\eta(C))$ or, equivalently, whenever $(\rho \sqcup \eta)^*$ computed in $uco(\rho(C))$ is the same closure as $\rho \sqcup \eta^*$.

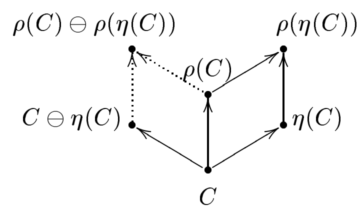

Figure 2: Basic abstraction structure.

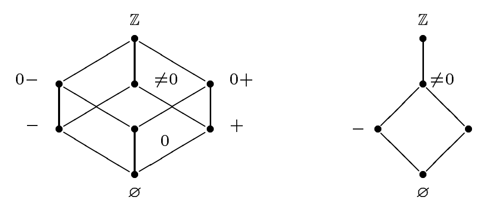

Figure 3: The $Sign$ and $Nzero$ domains.

EXAMPLE 5.1. *Note that it is not always the case that ρ is an $\ominus$-morphism. Let Sign be the domain for sign analysis of integer variables in Fig. 3 and $\rho(Sign) = \{\mathbb{Z}, 0-, -\}$. If $\eta(Sign) = \{\mathbb{Z}, 0-, \varnothing\}$, then $\rho(\eta(Sign)) = \{\mathbb{Z}, 0-\}$. Then we have $\rho(Sign \ominus \eta(Sign)) = \rho(\{\mathbb{Z}, 0+, \neq 0, +\}) = \{\mathbb{Z}\}$ while $\rho(Sign) \ominus \rho(\eta(Sign)) = \{\mathbb{Z}, -\}$.*

LEMMA 5.2. *Let $\rho, \eta \in uco(C)$. If $\rho \circ \eta = \eta \circ \rho$ and $\rho \circ \eta^* = \eta^* \circ \rho$ then $(\rho \sqcup \eta^*) \sqsubseteq \rho \ominus (\rho \sqcup \eta)$.*

PROOF. It is known that $\rho \circ \eta \in uco(C)$ iff $\rho \circ \eta = \eta \circ \rho = \rho \sqcup \eta$. It is immediate that $\rho \sqsubseteq (\rho \sqcup \eta) \sqcap (\rho \sqcup \eta^*)$. Moreover, by definition of domain complementation, for each $x \in C$: $\eta(x) \wedge \eta^*(x) = x$. Therefore, if $x \in \rho(C) \subseteq C$ we have

$$\begin{array}{rcl} x & = & \eta(x) \wedge \eta^*(x) \\ & = & \eta(\rho(x)) \wedge \eta^*(\rho(x)) \\ & = & (\rho \sqcup \eta)(x) \wedge (\rho \sqcup \eta^*)(x) \end{array}$$

Hence $\rho = (\rho \sqcup \eta) \sqcap (\rho \sqcup \eta^*)$, and by definition of pseudo-complementation in $uco(\rho(C))$: $(\rho \sqcup \eta^*) \sqsubseteq \rho \ominus (\rho \sqcup \eta)$. $\square$

LEMMA 5.3. *Let $\rho, \eta \in uco(C)$. If $\eta \sqcup \eta^* = \{\top\}$ and $Mirr(\rho(C) \cap \eta^*(C)) \subseteq Mirr(\rho(C))$ then $\rho \ominus (\rho \sqcup \eta) \sqsubseteq (\rho \sqcup \eta^*)$.*

PROOF. Because $\eta \sqcup \eta^* = \{\top\}$ then $Mirr(\rho(C) \cap \eta^*(C)) \cap \eta(C) = \varnothing$. Therefore by Theorem 4.1 if $Mirr(\rho(C) \cap \eta^*(C)) \subseteq Mirr(\rho(C))$ then $\rho \ominus (\rho \sqcup \eta) \sqsubseteq (\rho \sqcup \eta^*)$. $\square$

LEMMA 5.4. *Let $\rho, \eta \in uco(C)$. If $\rho \circ \eta = \eta \circ \rho$ and $\eta \sqcup \eta^* = \{\top\}$ then $\rho \circ \eta^* = \eta^* \circ \rho$.*

PROOF. For any $x \in C$ we have:

$$\begin{array}{rcl} \rho(\eta^*(x)) & = & \eta(\rho(\eta^*(x))) \wedge \eta^*(\rho(\eta^*(x))) \\ & = & \rho(\eta(\eta^*(x))) \wedge \eta^*(\rho(\eta^*(x))) \\ & = & \top \wedge \eta^*(\rho(\eta^*(x))) \\ & = & \eta^*(\rho(\eta^*(x))) \end{array}$$

Therefore, $\rho(\eta^*(x)) = \rho(\eta^*(\rho(\eta^*(x))))$, namely $\rho \circ \eta^*$ is monotone, idempotent and extensive, i.e., $\rho \circ \eta^* \in uco(C)$ which is true iff $\rho \circ \eta^* = \eta^* \circ \rho$. $\square$

It is worth noting that the structure of meet-irreducible elements is not always preserved by abstraction. This is a consequence of the structure of abstract domains which are Moore families: $\rho(Mirr(C)) \subseteq Mirr(\rho(C))$ and the inverse inclusion does not hold in general, as shown in Example 5.1. The following result gives two sufficient conditions such that ρ commutes with $\ominus$. These conditions are both formulated in terms of the structure of the meet-irreducible elements of abstract domains.

THEOREM 5.5. *Let $\rho, \eta \in uco(C)$ such that $\rho \circ \eta = \eta \circ \rho$.*

(i) If $\eta \sqcup \eta^* = \{\top\}$ and $Mirr(\rho(C) \cap \eta^*(C)) \subseteq Mirr(\rho(C))$ then $\rho(C \ominus \eta(C)) = \rho(C) \ominus \rho(\eta(C))$.

(ii) If $Mirr(\rho(C)) = \rho(Mirr(C))$ then $\rho(C \ominus \eta(C)) = \rho(C) \ominus \rho(\eta(C))$.

PROOF. *(i).* Immediate by Lemma 5.2, 5.3, and 5.4.
(ii). We prove $\rho(Mirr(C)) \smallsetminus \rho(\eta(C)) = \rho(Mirr(C) \smallsetminus \eta(C))$. We know that for any function f: $f(A) \smallsetminus f(B) \subseteq f(A \smallsetminus B)$ so the inclusion $\subseteq$ is valid. We now prove that:

$$x \in \rho(Mirr(C) \smallsetminus \eta(C)) \;\Rightarrow\; x \notin \rho(\eta(C))$$

Because $\rho \circ \eta = \eta \circ \rho$:

$$\begin{array}{rl} x \in \rho(\eta(C)) & \Rightarrow\; x \in \eta(C) \\ & \Rightarrow\; x \notin Mirr(C) \smallsetminus \eta(C) \\ & \Rightarrow\; x = \rho(x) \notin \rho(Mirr(C) \smallsetminus \eta(C)). \end{array}$$

Indeed, if $x \in \rho(Mirr(C) \smallsetminus \eta(C))$ then $x \in Mirr(C) \smallsetminus \eta(C)$ which is absurd because $x \in \rho(C) \cap \eta(C)$. Moreover we have $\rho(Mirr(C) \smallsetminus \eta(C)) \subseteq \rho(Mirr(C))$ and this fact together with what we proved above implies that

$$\rho(Mirr(C) \smallsetminus \eta(C)) \subseteq \rho(Mirr(C)) \smallsetminus \rho(\eta(C))$$

So we have the equality.
By the argument above this implies the following equalities:

$$\begin{array}{rcl} Mirr(\rho(C) \ominus \rho(\eta(C))) & = & Mirr(\rho(C)) \smallsetminus \rho(\eta(C)) \\ & = & \rho(Mirr(C)) \smallsetminus \rho(\eta(C)) \\ & = & \rho(Mirr(C) \smallsetminus \eta(C)) \end{array}$$

from which follows the thesis. $\square$

The following example shows that the converse of Theorem 5.5-(ii) does not hold in general.

EXAMPLE 5.6. *Let Sign and $\rho(Sign) = Nzero$ be the domains in Fig. 3 for sign and non-zero analysis of integer variables. If $\eta(Sign) = \{\mathbb{Z}, 0-, -\}$ then $\rho(\eta(Sign)) = \{\mathbb{Z}, -\}$. In this case we have that $\rho(Sign \ominus \eta(Sign)) = \rho(\{\mathbb{Z}, 0+, \neq 0, +\}) = \{\mathbb{Z}, \neq 0, +\} = Nzero \ominus \{\mathbb{Z}, -\}$ while $\rho(Mirr(Sign)) = \{\neq 0\}$ and $Mirr(Nzero) = \{\neq 0, +, -\}$.*

6. COMPLEMENTING SEMANTICS

In order to better understand how complementation factorizes a semantics, we need to characterize the reduced product of semantics. Let $Program$ be the collection of all well-formed programs in a programming language. If C is some domain, e.g. execution traces, functions, sets of states, etc. ([8]), and $[\![\cdot]\!] : Program \to C$ is the semantics assigning with each program $P \in Program$ its meaning in C, then for any $\rho \in uco(C)$: $[\![\cdot]\!]_\rho : Program \to \rho(C)$ is the so called abstract semantics, and it is $[\![P]\!]_\rho \stackrel{\text{def}}{=} \rho([\![P]\!])$. The following result [20] formally expresses the intuition that the reduced product semantics corresponds precisely to the logical conjunction of the observables associated with each semantics in the product.

THEOREM 6.1. *If $P, Q \in Program$ and $\{A_i\}_{i \in I} \subseteq uco(C)$ then $[\![P]\!]_{\sqcap_{i \in I} A_i} = [\![Q]\!]_{\sqcap_{i \in I} A_i}$ iff $\forall i \in I.\ [\![P]\!]_{A_i} = [\![Q]\!]_{A_i}$.*

PROOF. Assume that $P, Q \in Program$.
($\Rightarrow$) Assume that $k \in I$. Since $\sqcap_{i \in I} \rho_i \sqsubseteq \rho_k$, then we have $[\![P]\!]_{\rho_k} = [\![Q]\!]_{\rho_k}$, as desired.
($\Leftarrow$) Since $\forall i \in I.\ \rho_i([\![P]\!]) = \rho_i([\![Q]\!])$, we have $\wedge_{i \in I} \rho_i([\![P]\!]) = \wedge_{i \in I} \rho_i([\![Q]\!])$ which proves the thesis. $\square$

Therefore, given any two semantics $X, A \in uco(C)$, such that $X \sqsubseteq A$, the complement semantics $[\![\cdot]\!]_{(X \ominus A)}$ is the most abstract semantics such that $[\![P]\!]_X = [\![Q]\!]_X$ iff $[\![P]\!]_A = [\![Q]\!]_A$ and $[\![P]\!]_{(X \ominus A)} = [\![Q]\!]_{(X \ominus A)}$. In practice it is always possible to define complements of semantics, since the hypothesis in Theorem 4.1 and 4.1 assuring their existence are extremely weak. In most cases in fact the domain of interpretation is a continuous or even algebraic lattice, or it is generated by its meet-irreducible elements.

6.1 Complementing trace-based semantics

In this section we prove that angelic and demonic semantics provide a conjunctive decomposition of natural semantics, and that angelic and infinite semantics form a minimal (most abstract) decomposition of natural semantics. This is now proved for the basic operational trace-based semantics only, which represents the bottom (most concrete) semantics in Cousot's hierarchy, see Fig. 1. We will generalize this construction to the whole hierachy in Sec. 6.2.

We consider the angelic/demonic/infinite closure operators on generic traces $Ang, Dem, Inf \in uco(\wp(\Sigma^{\vec{\infty}}), \subseteq)$ induced by, respectively, the angelic, demonic, and infinite abstractions of trace semantics: $Ang \stackrel{\text{def}}{=} \gamma^+ \circ \alpha^+$, $Dem \stackrel{\text{def}}{=} \gamma^\partial \circ \alpha^\partial$, and $Inf \stackrel{\text{def}}{=} \gamma^\omega \circ \alpha^\omega$. It is immediate to observe that for any $X \in \Sigma^{\vec{\infty}}$:

$$\begin{array}{rcl} Ang(X) & = & X \cup \Sigma^{\vec{\omega}} \\ Dem(X) & = & X \cup \left\{ \sigma \in \Sigma^{\vec{+}} \;\middle|\; \begin{array}{l} \delta \in X \cap \Sigma^{\vec{\omega}} \\ \delta_0 = \sigma_0 \end{array} \right\} \\ Inf(X) & = & X \cup \Sigma^{\vec{+}} \end{array}$$

We first consider the relational abstraction given by the closure $\rho^r \stackrel{\text{def}}{=} \gamma^r \circ \alpha^r$ (see Table 1) which is the following function:

$$\rho^r(X) = \left\{ \sigma \in \Sigma^{\vec{\infty}} \;\middle|\; \begin{array}{l} \delta \in X, \delta_0 = \sigma_0 \\ |\delta| = n < \omega \Rightarrow |\sigma| = k < \omega \wedge \\ \qquad\qquad\qquad \sigma_{k-1} = \delta_{n-1} \\ |\delta| = \omega \Rightarrow |\sigma| = \omega \end{array} \right\}$$

The relational semantics is $\mathcal{R} = \rho^r(\tau^{\vec{\infty}})$. We observe that $Mirr(\langle\wp(A), \subseteq\rangle) = \{X \,|\, \exists a \in A\,.\, X = A\smallsetminus\{a\}\}$ and that $\wp(A)$ is generated by its meet-irreducibles. We have the following results relating angelic, demonic, infinite and relational semantics.

LEMMA 6.2. *Let $\sigma \in \Sigma^{\vec{+}}$ and $\delta \in \Sigma^{\vec{\omega}}$. Then we have*

$$\begin{array}{ll} \Sigma^{\vec{\infty}} \setminus \{\sigma\} \in Ang & \Sigma^{\vec{\infty}} \setminus \{\delta\} \notin Ang \\ \Sigma^{\vec{\infty}} \setminus \{\sigma\} \notin Dem & \Sigma^{\vec{\infty}} \setminus \{\delta\} \in Dem \end{array}$$

PROOF. It is immediate by definition that if $\sigma \in \Sigma^{\vec{+}}$ and $\delta \in \Sigma^{\vec{\omega}}$ then: $Ang(\Sigma^{\vec{\infty}} \setminus \{\sigma\}) = (\Sigma^{\vec{\infty}} \setminus \{\sigma\}) \cup \Sigma^{\vec{\omega}} = \Sigma^{\vec{\infty}} \setminus \{\sigma\}$, $Ang(\Sigma^{\vec{\infty}} \setminus \{\delta\}) = (\Sigma^{\vec{\infty}} \setminus \{\delta\}) \cup \Sigma^{\vec{\omega}} = \Sigma^{\vec{\infty}}$. The case for Dem is analogous. $\square$

The following proposition is immediate by Lemma 6.2 and Theorem 4.1.

PROPOSITION 6.3. *$Dem^* = Ang$.*

However, angelic and demonic abstractions do not factorize natural semantics in most abstract factors. In fact, while the complement of demonic semantics is angelic, the converse does not hold. It is worth noting that $Ang^* \neq Dem$. In particular, for any finite trace $\sigma \in \Sigma^{\vec{+}}$, $Dem(\{\sigma\}) = \{\sigma\}$ while $Ang^*(\{\sigma\}) = \Sigma^{\vec{+}} \supset \{\sigma\}$.

PROPOSITION 6.4. *$Ang^* = Inf$ and $Inf^* = Ang$.*

PROOF. $Ang^* = \mathcal{M}(Mirr(\wp(\Sigma^{\vec{\infty}})) \smallsetminus Ang)$. Therefore we have that $X \in Mirr(Ang^*)$ iff there exists $\delta \in \Sigma^{\vec{\omega}}$ such that $X = \Sigma^{\vec{\infty}} \smallsetminus \{\delta\} = \Sigma^{\vec{+}} \cup (\Sigma^{\vec{\omega}} \smallsetminus \{\delta\})$. It is immediate therefore that $X \in Ang^*$ iff $X = X \cup \Sigma^{\vec{+}}$, namely $Ang^* = Inf$. The proof for $Inf^* = Ang$ is analogous. $\square$

It is worth noting that demonic semantics can be further factorized by the infinite one, namely $Dem \ominus Inf \neq Dem$.

LEMMA 6.5.

$$Mirr(Dem) = Mirr(Inf) \cup \left\{ X \subseteq \Sigma^{\vec{\infty}} \;\middle|\; \begin{array}{l} \exists \sigma \in \Sigma^{\vec{+}}\,.\, X = \Sigma^{\vec{\infty}} \smallsetminus (\Delta_\sigma \cup \{\sigma\}), \\ \Delta_\sigma = \{\delta \in \Sigma^{\vec{\omega}} \mid \delta_0 = \sigma_0\} \end{array} \right\}$$

PROOF. We denote $Mirr(Dem)$ by M and we prove the two directions of the equality separately. In the following if $X \subseteq \Sigma^{\vec{\infty}}$ then $X^+ \stackrel{\text{def}}{=} X \cap \Sigma^{\vec{+}}$ and $X^\omega \stackrel{\text{def}}{=} X \cap \Sigma^{\vec{\omega}}$.
We first prove that if $X \in M$ then X is meet-irreducible in Dem. If $X \in Mirr(Inf)$ then it is immediate that X is meet-irreducible in Dem. Now, if $X = \Sigma^{\vec{\infty}} \smallsetminus (\Delta_\sigma \cup \{\sigma\})$ we have several cases. We can take $A_1, A_2 \in Dem$ such that $A_1 \cap A_2 = X$, then we have that $\Sigma^{\vec{\omega}} \smallsetminus \Delta_\sigma \subseteq A_i^\omega$, and $\Sigma^{\vec{+}} \smallsetminus \{\sigma\} \subseteq A_i^+$, for $i = 1, 2$. So we have only these possible cases : $A_i^\omega = \Sigma^{\vec{\omega}} \smallsetminus \Delta_\sigma$, $A_i^\omega = \Sigma^{\vec{\omega}} \smallsetminus \Delta_0$ where $\Delta_0 \subset \Delta_\sigma$ or $A_i^\omega = \Sigma^{\vec{\omega}}$; on the other hand we have $A_i^+ = \Sigma^{\vec{+}} \smallsetminus \{\sigma\}$ or $A_i^+ = \Sigma^{\vec{+}}$. Now we prove that in all these cases either $A_i = X$ or we have an absurd. We consider $A_1^\omega = \Sigma^{\vec{\omega}} \smallsetminus \Delta_\sigma$. Then $A_1^+ = \Sigma^{\vec{+}}$ (in the other case we have $A_1 = X$) and we see what happens for A_2. $A_2^+ = \Sigma^{\vec{+}} \smallsetminus \{\sigma\}$, otherwise we have $A_1 \cap A_2 \neq X$, so if $A_2^\omega = \Sigma^{\vec{\omega}} \smallsetminus \Delta_\sigma$ then we have $A_2 = X$. Moreover if $A_2^\omega = \Sigma^{\vec{\omega}}$ or $A_2^\omega = \Sigma^{\vec{\omega}} \smallsetminus \Delta_0$ then $A_2 \notin Dem$. Now we consider $A_1^\omega = \Sigma^{\vec{\omega}}$. Then $A_1^+ = \Sigma^{\vec{+}}$ (in the other case $A_1 \notin Dem$) and we see what happens for A_2. If $A_2^\omega = \Sigma^{\vec{\omega}}$ or $A_2^+ = \Sigma^{\vec{+}}$, we have an absurd because $A_1 \cap A_2 \neq X$. So we consider $A_2^+ = \Sigma^{\vec{+}} \smallsetminus \{\sigma\}$ and if $A_2^\omega = \Sigma^{\vec{\omega}} \smallsetminus \Delta_\sigma$ then $A_2 = X$, else if $A_2^\omega = \Sigma^{\vec{\omega}} \smallsetminus \Delta_0$ then $A_2 \notin Dem$. Finally we consider $A_1^\omega = \Sigma^{\vec{\omega}} \smallsetminus \Delta_0$. We have $A_1^+ = \Sigma^{\vec{+}}$, otherwise $A_1 \notin Dem$. Then $A_2^+ = \Sigma^{\vec{+}} \smallsetminus \{\sigma\}$, otherwise $A_1 \cap A_2 \neq X$. Moreover, if $A_2^\omega = \Sigma^{\vec{\omega}}$ then $A_1 \cap A_2 \notin Dem$, if $A_2^\omega = \Sigma^{\vec{\omega}} \smallsetminus \Delta_0$ then $A_1 \cap A_2 \neq X$, and if $A_2^\omega = \Sigma^{\vec{\omega}} \smallsetminus \Delta_\sigma$ then $A_2 = X$.
Now we have to prove that if X is meet-irreducible then it belongs to M. We suppose that $X \notin M$. Then there exists $D_0 \subseteq \Sigma^{\vec{\omega}}$ with $D_0 \supset \Delta_\sigma$ such that $X^\omega = \Sigma^{\vec{\omega}} \smallsetminus D_0$ or there exists $D_1 \subseteq \Sigma^{\vec{+}}$ with $|D_1| > 1$ such that $X^+ = \Sigma^{\vec{+}} \smallsetminus D_1$ or $X \notin Mirr(Inf)$. Now we distinguish two cases; if $X^+ = \Sigma^{\vec{+}}$ then we have to prove that $X \in Mirr(Inf)$ and this is immediate. So we consider $X^+ \subset \Sigma^{\vec{+}}$. Now we consider $X^+ = \Sigma^{\vec{+}} \smallsetminus D_1$ and we define $D_1' \stackrel{\text{def}}{=} D_1 \smallsetminus \{\sigma'\}$, with $\sigma' \in D_1$. Then we can take $A = X^\omega \cup (\Sigma^{\vec{+}} \smallsetminus D_1')$ and $B = X^\omega \cup (\Sigma^{\vec{+}} \smallsetminus \{\sigma'\})$. In this case $A \cap B = X$ but $A \neq X$ and $B \neq X$, which is absurd. So $X^+ = \Sigma^{\vec{+}} \smallsetminus \{\sigma\}$. Now we consider $X^\omega = \Sigma^{\vec{\omega}} \smallsetminus D_0$, we define $D_0' \stackrel{\text{def}}{=} D_0 \smallsetminus \{\delta'\}$ with $\delta' \notin \Delta_\sigma$ and $\delta' \in D_0$. Then we can take $A = \Sigma^{\vec{\infty}} \smallsetminus (D_0' \cup \{\sigma\})$ while $B = \Sigma^{\vec{\infty}} \smallsetminus \{\delta'\}$, so we have that $A \cap B = X$ but $A \neq X$ and $B \neq X$, which is absurd. $\square$

The following closure operator, here called *pseudo-demonic*, characterizes the new complementary semantics defined by $Dem \ominus Inf$: $Dem^\varrho \in uco(Dem(\wp(\Sigma^{\vec{\infty}})))$ such that

$$Dem^\varrho \stackrel{\text{def}}{=} \lambda X\,.\, X \cup \left\{ \delta \in \Sigma^{\vec{\omega}} \;\middle|\; chaos(\delta) \subseteq X^+ \right\}$$

The idea here is that this new semantics is unable to distinguish whether a chaotic set of traces is a consequence of the existence of an infinite trace in $\tau^{\vec{\infty}}$. This abstraction is achieved by enhancing any set of traces X with all the infinite traces $\delta \in \Sigma^{\vec{\omega}}$ whenever the chaos generated by δ, namely $chaos(\delta) \stackrel{\text{def}}{=} \{\sigma \in \Sigma^{\vec{+}} \mid \sigma_0 = \delta_0\}$, is contained in X.
The following result is straightforward by Theorem 4.1 and by Lemma 6.5. As expected, this new semantics which is $Dem^\varrho(Dem(\tau^{\vec{\infty}}))$ is unable to observe infinite behavior, as well as in the angelic case, whenever chaotic information is computed in $\tau^{\vec{\infty}}$. In particular $(Dem \ominus Inf) \circ Dem$ is unable to factorize the basic trace semantics. Indeed, for any $X \in Mirr(\wp(\Sigma^{\vec{\infty}}))$ we have $(Dem \ominus Inf)(Dem(X)) = \Sigma^{\vec{\infty}}$.

THEOREM 6.6. *$Dem \ominus Inf = Dem^\varrho$, $Dem \ominus Dem^\varrho = Inf$, and $(Dem^\varrho \circ Dem)^* = \lambda x.\, x$.*

The angelic abstraction is fully complementary with respect to infinite and demonic ones, namely they share only the top element in $\wp(\Sigma^{\vec{\infty}})$ which is the set of all sequences $\Sigma^{\vec{\infty}}$.

PROPOSITION 6.7.
$Ang \sqcup Dem = Ang \sqcup Inf = Dem^\varrho \sqcup Inf = \Sigma^{\vec{\infty}}$.

In order to characterize the symmetric structure of the observables at the level of trace-based and relational semantics we consider Theorem 5.5, and we prove that angelic, demonic, pseudo-demonic, and infinite abstractions all provide commuting closures with respect to the relational abstraction.

PROPOSITION 6.8.

1. $\rho^r \circ Ang = Ang \circ \rho^r$
2. $\rho^r \circ Dem = Dem \circ \rho^r$
3. $\rho^r \circ Inf = Inf \circ \rho^r$
4. $\rho^r \circ Dem^{\varrho} = Dem^{\varrho} \circ \rho^r$

PROOF. It is easy to prove that ρ^r is additive. Therefore

$$\begin{array}{rcl} \rho^r(Ang(X)) & = & \rho^r(X \cup \Sigma^{\vec{\omega}}) \\ & = & \rho^r(X) \cup \rho^r(\Sigma^{\vec{\omega}}) \\ & = & \rho^r(X) \cup \Sigma^{\vec{\omega}} \\ & = & Ang(\rho^r(X)). \end{array}$$

We prove that $\rho^r \circ Dem = Dem \circ \rho^r$. By definition and additivity of ρ^r we have $\rho^r(Dem(X)) = \rho^r(X) \cup \rho^r(\{\sigma \in \Sigma^{\vec{+}} | \delta \in X \cap \Sigma^{\vec{\omega}}, \delta_0 = \sigma_0\})$ and $Dem(\rho^r(X)) = \rho^r(X) \cup \{\sigma \in \Sigma^{\vec{+}} | \delta \in \rho^r(X) \cap \Sigma^{\vec{\omega}}, \delta_0 = \sigma_0\}$. Note that $\theta \in \rho^r(\{\sigma \in \Sigma^{\vec{+}} | \delta \in X \cap \Sigma^{\vec{\omega}}, \delta_0 = \sigma_0\})$ iff $\theta \in \Sigma^{\vec{+}}$, $\sigma \in \Sigma^{\vec{+}}$, $\delta \in X \cap \Sigma^{\vec{\omega}}$, and $\delta_0 = \sigma_0$, $\sigma_0 = \theta_0$ and $\theta_{n-1} = \sigma_{k-1}$, with $|\theta| = n$ and $|\sigma| = k$. Therefore $\theta \in \{\sigma \in \Sigma^{\vec{+}} | \delta \in X \cap \Sigma^{\vec{\omega}}, \delta_0 = \sigma_0\}$. This implies that $\rho^r(\{\sigma \in \Sigma^{\vec{+}} | \delta \in X \cap \Sigma^{\vec{\omega}}, \delta_0 = \sigma_0\}) = \{\sigma \in \Sigma^{\vec{+}} | \delta \in X \cap \Sigma^{\vec{\omega}}, \delta_0 = \sigma_0\}$. Moreover if θ is such that $\delta \in \rho^r(X) \cap \Sigma^{\vec{\omega}}$ and $\delta_0 = \theta_0$, then by definition of ρ^r, there exists $\delta' \in X \cap \Sigma^{\vec{\omega}}$ such that $\delta_0 = \delta'_0 = \theta_0$. This implies that $\theta \in \rho^r(\{\sigma \in \Sigma^{\vec{+}} | \delta \in X \cap \Sigma^{\vec{\omega}}, \delta_0 = \sigma_0\})$ iff $\theta \in \{\sigma \in \Sigma^{\vec{+}} | \delta \in \rho^r(X) \cap \Sigma^{\vec{\omega}}, \delta_0 = \sigma_0\}$. The equality $\rho^r \circ Inf = Inf \circ \rho^r$ follows by Lemma 5.4. The case for Dem^{ϱ} is analogous because $\rho^r(\{\delta \in \Sigma^{\vec{\omega}} \mid chaos(\delta) \subseteq X\}) = \{\delta \in \Sigma^{\vec{\omega}} \mid chaos(\delta) \subseteq \rho^r(X)\}$. $\square$

It is worth noting that the relational semantics does not factorize the trace-based one, being too abstract, i.e., $(\rho^r)^* = \lambda x.x$ or equivalently $\tau^{\vec{\infty}} \ominus \mathcal{R} = \tau^{\vec{\infty}}$. This does not allows us to the use Theorem 5.5-(ii) for relating the observables at the level of trace-based and relational semantics styles in Cousot's hierarchy. Instead we can use Theorem 5.5-(i), as shown below.

PROPOSITION 6.9.

1. $Mirr(\rho^r(\wp(\Sigma^{\vec{\infty}})) \cap Ang(\wp(\Sigma^{\vec{\infty}}))) \subseteq Mirr(\rho^r(\wp(\Sigma^{\vec{\infty}})))$
2. $Mirr(\rho^r(\wp(\Sigma^{\vec{\infty}})) \cap Inf(\wp(\Sigma^{\vec{\infty}}))) \subseteq Mirr(\rho^r(\wp(\Sigma^{\vec{\infty}})))$
3. $Mirr(\rho^r(Dem(\wp(\Sigma^{\vec{\infty}}))) \cap Dem^{\varrho}(\wp(\Sigma^{\vec{\infty}}))) \subseteq Mirr(\rho^r(Dem(\wp(\Sigma^{\vec{\infty}}))))$

Therefore, it is immediate by Theorem 5.5-(i) that:

$$\begin{array}{rcl} \mathcal{R} & = & Ang(\mathcal{R}) \sqcap Dem(\mathcal{R}) = Ang(\mathcal{R}) \sqcap Inf(\mathcal{R}) \\ Dem(\mathcal{R}) & = & Dem^{\varrho}(\mathcal{R}) \sqcap Inf(\mathcal{R}) \\ Inf(\mathcal{R}) & = & Dem(\mathcal{R}) \ominus Dem^{\varrho}(\mathcal{R}) \\ Dem^{\varrho}(\mathcal{R}) & = & Dem(\mathcal{R}) \ominus Inf(\mathcal{R}) \\ Inf(\mathcal{R}) & = & \mathcal{R} \ominus Ang(\mathcal{R}) \\ Ang(\mathcal{R}) & = & \mathcal{R} \ominus Dem(\mathcal{R}). \end{array}$$

6.2 Complementing all over the hierarchy

In this section we prove that the basic pattern of observables given by angelic, demonic, pseudo-demonic, and infinite can always be characterized by domain complementation for any semantics style in Cousot's hierarchy. The following lemma characterizes the meet-irreducible elements in natural nondeterministic denotational semantics $\mathcal{D}$. This semantics is isomorphic to the basic natural relational semantics $\mathcal{R}$ [8].

LEMMA 6.10.

$$\begin{array}{l} Mirr(\Sigma \longrightarrow \wp(\Sigma_\perp), \sqsubseteq) = \\ \quad = \left\{ f \;\middle|\; \begin{array}{l} \exists \overline{s} \in \Sigma, \exists s' \in \Sigma_\perp \,.\, f(\overline{s}) = \Sigma_\perp \smallsetminus \{s'\} \wedge \\ \forall s \in \Sigma \smallsetminus \{\overline{s}\} \,.\, f(s) = \Sigma_\perp \end{array} \right\} \end{array}$$

PROOF. Let A and B be sets. We prove the two implications of the equality separately.

($\Rightarrow$) Let f be a meet-irreducible element in $A \longrightarrow \wp(B)$, then for all $g, h \in A \longrightarrow \wp(B)$ we have that if $f = g \sqcap h$ then either $g = f$ or $h = f$. Assume $f \notin M$. By definition of M, this implies that either (1) $\exists \overline{a} \in A \,.\, f(\overline{a}) = B \smallsetminus D$, $|D| > 1$ or (2)$\exists a \neq \overline{a} \,.\, f(s) \subset B$. We consider the two cases separately

(1) Let $b_0 \in D$ and $D_0 = D \smallsetminus \{b_0\}$ with $|D_0| \geq 1$, then we can define the functions

$$h(a) = \begin{cases} f(a) & \text{if } a \neq \overline{a} \\ B \smallsetminus \{b_0\} & \text{otherwise} \end{cases}$$

$$g(a) = \begin{cases} f(a) & \text{if } a \neq \overline{a} \\ B \smallsetminus D_0 & \text{otherwise} \end{cases}$$

Note that $f \sqsubset h$ and $f \sqsubset g$ because $f(\overline{a}) \subset h(\overline{a})$ and $f(\overline{a}) \subset g(\overline{a})$, while for the other elements the functions coincide. This implies that $f \sqsubseteq h \sqcap g$. We prove the other inclusion. Foe each $a \neq \overline{a}$ we have that $h(a) = g(a) = f(a)$; in particular $f(a) = h(a) \cap g(a)$, while $h(\overline{a}) = B \smallsetminus \{b_0\}$ and $g(\overline{a}) = B \smallsetminus D_0$. Therefore $h(\overline{a}) \cap g(\overline{a}) = B \smallsetminus (D_0 \cup \{b_0\}) = B \smallsetminus D = f(\overline{a})$. Hence for all $a \in A$ we have that $f(a) = h(a) \cap g(a)$, with $f \neq h$ and $f \neq g$, but this is absurd from the hypothesis.

(2) Let $\overline{a} \in A$ such that $f(\overline{a}) = B \smallsetminus \{b\}$, and let $a_0 \in A \smallsetminus \{\overline{a}\}$ such that $f(a_0) \subset B$. If $f(a_0) = B \smallsetminus D$ with $|D| > 1$ then we fall into the previous case; therefore $f(a_0) = B \smallsetminus \{b_0\}$. We define the following functions

$$h(a) = \begin{cases} f(a) & \text{if } a \neq \overline{a} \\ B & \text{otherwise} \end{cases}$$

$$g(a) = \begin{cases} f(a) & \text{if } a \neq a_0 \\ B & \text{otherwise} \end{cases}$$

In this case we have that $f \sqsubset g$ and $f \sqsubset h$ because $f(\overline{a}) \subset B$ and $f(a_0) \subset B$. Observe that for each $a \neq \overline{a}, a_0$ we have $f(a) = g(a) = h(a) = g(a) \cap h(a)$; while $f(\overline{a}) = g(\overline{a}) \subset h(\overline{a})$ and therefore $g(\overline{a}) \cap h(\overline{a}) = g(\overline{a}) = f(\overline{a})$. In the same way $f(a_0) = h(a_0) \subset g(a_0)$ and therefore $g(a_0) \cap h(a_0) = h(a_0) = f(a_0)$.
So for all $a \in A$ we have $f(a) = g(a) \cap h(a)$ but $f \neq g$ and $f \neq h$, and this is absurd from the hypothesis.

($\Leftarrow$) Let $f \in M$. Then there exists $\overline{a} \in A$ and $b \in B$ such that $f(\overline{a}) = B \setminus \{b\}$. Let g and h be such that $f = g \sqcap h$. Then $\forall a \in A \,.\, f(a) = g(a) \cap h(a)$. This means that $\forall a \in A \,.\, f(a) \subseteq g(a)$ and $\forall a \in A \,.\, f(a) \subseteq h(a)$. We know that if $a \neq \overline{a}$ then $f(a) = B \subseteq g(a)$, but this implies that for these a's, $g(a) = B$. In the same way we can say that for these a's, $h(a) = B$. We consider now $\overline{a}$. Then we have:

$$\begin{array}{l} g(\overline{a}) \supseteq B \setminus \{b\} \ \Rightarrow \\ \qquad \text{if } g(\overline{a}) = B \setminus \{b\} \text{ then } g = f \\ \qquad \text{if } h(\overline{a}) = B \text{ then } g(\overline{a}) \cap h(\overline{a}) = B \neq f(\overline{a}) \end{array}$$

which is absurd. So $h(\overline{a}) = B \setminus \{b\}$ and then $h = f$. All these facts together mean that f is meet-irreducible.

This concludes the proof. $\square$

The following result proves that all semantics in Cousot's hierarchy satisfy the hypothesis of Theorem 5.5-(ii). In the following we let $Mirr_{\mathcal{R}} \stackrel{\text{def}}{=} Mirr(\langle \wp(\Sigma \times \Sigma_\perp), \subseteq \rangle)$.

LEMMA 6.11.

$$\begin{array}{lcr} Mirr(\langle \Sigma \longrightarrow \wp(\Sigma_\perp), \sqsubseteq \rangle) & = & \alpha^{\blacktriangleright}(Mirr_{\mathcal{R}}) \\ Mirr(\langle \wp(\Sigma_\perp) \stackrel{coa}{\longrightarrow} \wp(\Sigma), \sqsupseteq \rangle) & = & \alpha^{gwp}(\alpha^{\blacktriangleright}(Mirr_{\mathcal{R}})) \\ Mirr(\langle \wp(\Sigma) \otimes \wp(\Sigma_\perp), \supseteq \rangle) & = & \alpha^{gH}(\alpha^{gwp}(\alpha^{\blacktriangleright}(Mirr_{\mathcal{R}}))) \end{array}$$

PROOF. We just prove that $Mirr(\langle \Sigma \longrightarrow \wp(\Sigma_\perp), \sqsubseteq \rangle) = \alpha^{\blacktriangleright}(Mirr_{\mathcal{R}})$. We know that $\alpha^{\blacktriangleright}(X) = \lambda s.\{s' \mid \langle s, s' \rangle \in X\}$. Now we prove the two implications of the equality separately. Let $X \in Mirr_{\mathcal{R}}$ then

$$\begin{array}{rcl} \alpha^{\blacktriangleright}(X) & = & \alpha^{\blacktriangleright}((\Sigma \times \Sigma_\perp) \setminus \{\langle s_0, s_1 \rangle\}) \\ & = & \begin{cases} \lambda s.\Sigma_\perp & \text{if } s \neq s_0 \\ \lambda s.\Sigma_\perp \setminus \{s_1\} & \text{otherwise} \end{cases} \end{array}$$

But this implies that $\alpha^{\blacktriangleright}(X) \in Mirr(\langle \Sigma \longrightarrow \wp(\Sigma_\perp), \sqsubseteq \rangle)$. Let now $f \in Mirr(\langle \Sigma \longrightarrow \wp(\Sigma_\perp), \sqsubseteq \rangle)$ then it exists $s_0 \in \Sigma$ and $s_1 \in \Sigma_\perp$ such that $f(s_0) = \Sigma_\perp \setminus \{s_1\}$ and for all $s \neq s_0$ we have $f(s) = \Sigma_\perp$. Then we can take $X = \Sigma \times \Sigma_\perp \setminus \{\langle s_0, s_1 \rangle\} \in Mirr_{\mathcal{R}}$ such that $f = \alpha^{\blacktriangleright}(X)$. $\square$

The lemma above together with Theorem 5.5-(ii) implies the following result relating program semantics at different levels of abstraction, see Fig. 1. In this case we extend the scope of $\ominus$ from closures to semantics in the obvious way: Let $\mathcal{A} = \rho^{\mathcal{A}}(\tau^{\vec{\infty}})$ and $\mathcal{B} = \eta^{\mathcal{B}}(\tau^{\vec{\infty}})$ then $\mathcal{A} \ominus \mathcal{B} = (\rho \ominus \eta)(\tau^{\vec{\infty}})$. The semantics not originally included in Cousot's hierarchy are represented in Fig. 1 with dashed lines and empty points.

THEOREM 6.12. *In Cousot's hierarchy of semantics we have* $\tau^{\vec{\infty}} \ominus \tau^{\vec{+}} = \tau^{\vec{\omega}}$ *and* $\tau^{\vec{\infty}} \ominus \tau^{\vec{\omega}} = \tau^{\vec{\infty}} \ominus \tau^{\vec{\partial}} = \tau^{\vec{+}}$*. Moreover:*

$$\begin{array}{ll} \mathcal{R} \ominus \mathcal{R}^{+} = \mathcal{R}^{\omega} & \mathcal{D} \ominus \mathcal{D}^{+} = \mathcal{D}^{\omega} \\ \mathcal{R} \ominus \mathcal{R}^{\partial} = \mathcal{R} \ominus \mathcal{R}^{\omega} = \mathcal{R}^{+} & \mathcal{D} \ominus \mathcal{D}^{\partial} = \mathcal{D} \ominus \mathcal{D}^{\omega} = \mathcal{D}^{+} \\ \mathcal{R}^{\partial} \ominus \mathcal{R}^{\varrho} = \mathcal{R}^{\omega} & \mathcal{D}^{\partial} \ominus \mathcal{D}^{\varrho} = \mathcal{D}^{\omega} \\ \mathcal{R}^{\partial} \ominus \mathcal{R}^{\omega} = \mathcal{R}^{\varrho} & \mathcal{D}^{\partial} \ominus \mathcal{D}^{\omega} = \mathcal{D}^{\varrho} \end{array}$$

$$\begin{array}{ll} gwp \ominus wlp = wp^{\omega} & g\mathcal{H} \ominus p\mathcal{H} = t\mathcal{H}^{\omega} \\ gwp \ominus wp = gwp \ominus wp^{\omega} = wlp & g\mathcal{H} \ominus t\mathcal{H} = g\mathcal{H} \ominus t\mathcal{H}^{\omega} = p\mathcal{H} \\ wp \ominus wp^{\varrho} = wp^{\omega} & t\mathcal{H} \ominus t\mathcal{H}^{\varrho} = t\mathcal{H}^{\omega} \\ wp \ominus wp^{\omega} = wp^{\varrho} & t\mathcal{H} \ominus t\mathcal{H}^{\omega} = t\mathcal{H}^{\varrho} \end{array}$$

6.3 Complements in the computational order

We consider here the lifted computational order $\sqsubseteq^{\infty}$ on the trace-based semantics $\wp(\Sigma^{\vec{\infty}})$. We prove that most of the above results still hold for the computational order $\sqsubseteq^{\infty}$, therefore they can be used to characterize complementary semantics defined by combining both inductive and co-inductive definitions. In this case it is possible to characterize the complement of the angelic relational semantics which is again the infinite relational semantics and, conversely, the complement of the infinite semantics which is the angelic one. This is not the case for the demonic semantics which is not an upper closure operator on $\langle \wp(\Sigma^{\vec{\infty}}), \sqsubseteq^{\infty} \rangle$, (cf. [8]).

THEOREM 6.13. *Let* $M_\Sigma \stackrel{\text{def}}{=} \{\Sigma^{\vec{+}} \cup \{\delta\} \mid \delta \in \Sigma^{\vec{\omega}}\}$*, then* $Mirr(\langle \wp(\Sigma^{\vec{\infty}}), \sqsubseteq^{\infty} \rangle) = Mirr(\langle \wp(\Sigma^{\vec{+}}), \subseteq \rangle) \cup M_\Sigma$.

PROOF. We prove the two directions of the equality separately.

We first prove that if $X \in Mirr(\wp(\Sigma^{\vec{+}})) \cup M_\Sigma$ then X is irreducible in $\langle \wp(\Sigma^{\vec{\infty}}), \sqsubseteq^{\infty} \rangle$. If $X \in Mirr(\wp(\Sigma^{\vec{+}}))$ then it is irreducible in $\langle \wp(\Sigma^{\vec{\infty}}), \sqsubseteq^{\infty} \rangle$ because it is irreducible in $\wp(\Sigma^{\vec{+}})$ and by definition of $\sqsubseteq^{\infty}$ it cannot be obtained as *glb* of sets containing infinite traces. Indeed let $X = A \sqcap^{\infty} B$, with $A, B \subseteq \Sigma^{\vec{\infty}}$ and suppose that there exists $\delta \in A^{\omega}$. Then $X \sqsubseteq^{\infty} A$ and this fact implies that $X^{+} \subseteq A^{+}$ and $X^{\omega} \supseteq A^{\omega}$. This last relation says that $\delta \in X^{\omega}$, but this is absurd because $X \in \wp(\Sigma^{\vec{+}})$ and so $X^{\omega} = \varnothing$.

Assume that $X \in M_\Sigma$, i.e., $X = \Sigma^{\vec{+}} \cup \{\delta\}$ for some $\delta \in \Sigma^{\vec{\omega}}$, and let A and B be as above, such that $A \sqcap^{\infty} B = X$. By definition of *glb* $X \sqsubseteq^{\infty} A, B$ and for each Y such that $Y \sqsubseteq^{\infty} A, B$ then $Y \sqsubseteq^{\infty} X$. So $X \sqsubseteq^{\infty} A$, which means that $\Sigma^{\vec{+}} \subseteq A^{+}$, and this implies that $A^{+} = \Sigma^{\vec{+}}$. On the other hand $\{\delta\} \supseteq A^{\omega}$ so $A^{\omega} = \varnothing$ or $A^{\omega} = \{\delta\}$. Then if $A^{\omega} = \{\delta\}$ it is obvious that $A = X$. We now suppose that $A^{\omega} = \varnothing$. Then we can do the same reasoning about B and so we can say that $B^{\omega} = \varnothing$ or $B^{\omega} = \{\delta\}$. If also $B^{\omega} = \varnothing$, then $A = B = \Sigma^{\vec{+}}$ which means that $A \sqcap^{\infty} B = A = \Sigma^{\vec{+}} \neq X$, but this is absurd for the hypothesis made. Therefore if $A^{\omega} = \varnothing$ then B is such that $B^{\omega} = \{\delta\}$, and in this way we have that $B = X$, proving that X is meet-irreducible.

We now prove that if X is meet-irreducible then we have $X \in Mirr(\wp(\Sigma^{\vec{+}})) \cup M_\Sigma$. We suppose that $X \notin Mirr(\wp(\Sigma^{\vec{+}})) \cup M_\Sigma$, i.e., $X \notin Mirr(\wp(\Sigma^{\vec{+}}))$ and $X \notin M_\Sigma$. If $X^{\omega} = \varnothing$ then $X \subseteq \Sigma^{\vec{+}}$ and from the hypothesis of irreducibility of X, X belongs to $Mirr(\wp(\Sigma^{\vec{+}}))$, but this is absurd from the hypothesis made. Therefore let $X = X^{+} \cup X^{\omega}$ be such that $X^{\omega} \neq \varnothing$. Because $X \notin M_\Sigma$ then we have that either $X^{+} \subset \Sigma^{\vec{+}}$ or $|X^{\omega}| > 1$.

- Assume $X^{+} \subset \Sigma^{\vec{+}}$ then we can take $A \stackrel{\text{def}}{=} X^{+} \sqsupset^{\infty} X$ which holds because $X^{\omega} \supset \varnothing$; and $B \stackrel{\text{def}}{=} \Sigma^{\vec{+}} \cup X^{\omega} \sqsupset^{\infty} X$ which is true because $X^{+} \subset \Sigma^{\vec{+}}$. Then $X \sqsubseteq^{\infty} A$ and $X \sqsubseteq^{\infty} B$. We prove that X is the *glb* of A and B. Let $Y \sqsubseteq^{\infty} A, B$, then $Y^{+} \subseteq A^{+} = A = X^{+}$ and $Y^{\omega} \supseteq B^{\omega} = X^{\omega}$, so $Y \sqsubseteq^{\infty} X$, then by definition of *glb* $X = A \sqcap^{\infty} B$, but $X \neq A$ and $X \neq B$. The result is absurd for the hypothesis made for X. So if X is meet-irreducible then $X^{+} = \Sigma^{\vec{+}}$.
- Let $|X^{\omega}| > 1$ and let $\delta \in X^{\omega}$ and $D = X^{\omega} \setminus \{\delta\}$ with $D \neq \varnothing$. We take $A \stackrel{\text{def}}{=} X^{+} \cup \{\delta\} \sqsupset^{\infty} X$ which is true

because $X^\omega \supset \{\delta\}$. Moreover we take $B \stackrel{\text{def}}{=} X^+ \cup D \sqsupset^\infty X$ which is true because $X^\omega \supset D$. Then $X \sqsubseteq^\infty A$ and $X \sqsubseteq^\infty B$. Let $Y \sqsubseteq^\infty A, B$, then $Y^+ \subseteq A^+ = X^+$, $Y^\omega \supseteq A^\omega = \{\delta\}$ and $Y^\omega \supseteq B^\omega = D$. The last inequalities imply that $Y^\omega \supseteq D \cup \{\delta\} = X^\omega$, and therefore $Y \sqsubseteq^\infty X$. As in the previous case, we have that $X = A \sqcap^\infty B$ but $X \neq A$ and $X \neq B$; which is absurd for the hypothesis on X. Therefore if X is meet-irreducible then $|X^\omega| = 1$.

Hence if X is meet-irreducible then $X \in \mathit{Mirr}(\wp(\Sigma^{\vec{+}})) \cup M_\Sigma$, which concludes the proof. □

Cousot proved that α^+ and $\tilde{\gamma}^+ \stackrel{\text{def}}{=} \lambda x.\, x$ still define a Galois insertion between the natural trace-based semantics and its angelic counterpart under the lifted partial order $\sqsubseteq^\infty$. This means that $\mathit{Ang}^\infty \stackrel{\text{def}}{=} \alpha^+$ is a closure operator also in the computational order, i.e., $\mathit{Ang}^\infty \in \mathit{uco}(\langle \wp(\Sigma^{\vec{\infty}}), \sqsubseteq^\infty \rangle)$. The same holds for $\mathit{Inf} = \gamma^\omega \circ \alpha^\omega \in \mathit{uco}(\langle \wp(\Sigma^{\vec{\infty}}), \sqsubseteq^\infty \rangle)$. In this case the pseudo-complement of Ang^∞ and Inf coincide respectively with the pseudo-complement of the closure operator Ang and Inf with respect to the approximation order $\subseteq$, and they still provide a most abstract factorization of the trace-based semantics which is perfectly symmetric all over Cousot's hierarchy. Note that in the following all closure operators and constructions are considered in the complete lattice $\langle \wp(\Sigma^{\vec{\infty}}), \sqsubseteq^\infty, \sqcap^{\vec{\infty}}, \sqcup^{\vec{\infty}}, \sqcap^{\vec{\infty}}, \Sigma^{\vec{+}}, \Sigma^{\vec{\omega}} \rangle$. In particular $(\mathit{Ang}^\infty)^* = \mathit{Ang}^* = \mathit{Inf} = \mathcal{M}(M_\Sigma)$, $\mathit{Inf}^* = \mathit{Ang}^\infty$, $\mathit{Ang}^\infty \sqcup \mathit{Inf} = \Sigma^{\vec{+}}$, and $\rho^r \circ \mathit{Ang}^\infty = \mathit{Ang}^\infty \circ \rho^r$. Therefore, also Theorem 6.12 holds in the computational order $\sqsubseteq^\infty$ for angelic and infinite semantics.

Theorem 6.14. *In Cousot's hierarchy with computational order $\sqsubseteq^\infty$ we have $\tau^{\vec{\infty}} \ominus \tau^{\vec{+}} = \tau^{\vec{\omega}}$ and $\tau^{\vec{\infty}} \ominus \tau^{\vec{\omega}} = \tau^{\vec{+}}$. Moreover we have:*

$$
\begin{array}{ll}
\mathcal{R} \ominus \mathcal{R}^+ = \mathcal{R}^\omega & \mathcal{D} \ominus \mathcal{D}^+ = \mathcal{D}^\omega \\
\mathcal{R} \ominus \mathcal{R}^\omega = \mathcal{R}^+ & \mathcal{D} \ominus \mathcal{D}^\omega = \mathcal{D}^+ \\
 & \\
\mathit{gwp} \ominus \mathit{wlp} = \mathit{wp}^\omega & g\mathcal{H} \ominus p\mathcal{H} = t\mathcal{H}^\omega \\
\mathit{gwp} \ominus \mathit{wp}^\omega = \mathit{wlp} & g\mathcal{H} \ominus t\mathcal{H}^\omega = p\mathcal{H}
\end{array}
$$

7. AN EXAMPLE IN CCP

In this section we consider the case of *concurrent constraint programming* languages [28] and we derive its denotational closure-based [29] and axiomatic semantics [13] as an abstract interpretation of maximal traces of constraints in a transition system semantics.
Concurrent constraint programming (*ccp* for short) is a well known concurrent programming paradigm where processes interacts through a common store [28]. This leads to a computational model based on the notion of *store-as-constraint*. The main features of a *cc* process is to refine the store (tell-constraints) or synchronize itself with other processes (ask-constraints). The *ask-tell* paradigm, which is the basis of *ccp* languages, is an extension of constraint logic programming: In addition to satisfiability (tell), *entailment* (ask) is introduced. A store is a constraint representing the global state of the computation. Synchronization is achieved through *blocking ask*: A process is suspended when the store does not entail the ask constraint and it remains suspended until the store entails it.
The constraint system represents the basic algebraic notion behind *ccp*. The construction in [29] is an extension of Scott's partial information systems [1]. Informally, we have a denumerable set D of elementary assertions (containing distinct elements **1** and **0** representing the least informative assertion and the contradiction respectively) and a finite entailment relation $\vdash \subseteq \wp_f(D \times D)$. A *simple constraint system* is $\Sigma \stackrel{\text{def}}{=} \langle \wp(D), \dashv \rangle / _\sim$, which is a complete ω-algebraic lattice [1] where $X \sim Y$ iff $(X)^\dashv = (Y)^\dashv$, being $(X)^\dashv$ the entailment closure of a set of assertions X. An arbitrary element of Σ is called a *constraint*. Compact elements are called *finite constraints*, since they are equivalent to a finite subset of D. In order to treat the hiding operator of the language, Saraswat et al. [29] introduce a family of unary operations called *cylindrifications* see [23]). Intuitively, given a constraint c, the cylindrification operation $\exists_x(c)$ yields the constraint obtained by "projecting out" from c the information about the variable x. *Diagonal elements* (equational constraints between variables [23]) are considered as a way to provide parameter passing. Note that special variables (not accessible to the user) together with a suitable use of cylindrification and diagonal elements make variable renaming no longer needed [29].

Definition 7.1.
A constraint system $\langle \Sigma, \vdash, \mathit{false}, \mathit{true}, \wedge, V, \exists_x, d_{xy} \rangle$ *is an algebraic structure where:* $\langle \Sigma, \dashv \rangle$ *is a simple constraint system,* $\mathit{true} = [\mathbf{1}]_\sim$ *and* $\mathit{false} = [\mathbf{0}]_\sim$, $\wedge$ *is the glb,* V *is a denumerable set of variables, and* $\forall x, y \in V$, $\forall c, c' \in \Sigma$, *the operator* $\exists_x : \Sigma \to \Sigma$ *satisfies*

1. $c \vdash \exists_x c$
2. *if* $c \vdash c'$ *then* $\exists_x c \vdash \exists_x c'$
3. $\exists_x(c \wedge \exists_x c') = \exists_x c \wedge \exists_x c'$
4. $\exists_x(\exists_y c) = \exists_y(\exists_x c)$

$\forall x, y, z \in V$, $\forall c \in \Sigma$, *the diagonal element* d_{xy} *satisfies*

1. $d_{xx} = \mathit{true}$
2. *if* $z \neq x, y$ *then* $d_{xy} = \exists_z(d_{xz} \wedge d_{zy})$
3. *if* $x \neq y$ *then* $d_{xy} \wedge \exists_x(c \wedge d_{xy}) \vdash c$

The semantic operators of concurrent constraint languages are: elementary actions (**ask** and **tell**), hiding ($\exists$), parallel composition ($\|$), guarded nondeterministic choice ($\sum$) and recursion. The semantics of *ccp* programs well fits into Cousot's hierarchy being easily described as maximal *consistent* traces of a transition system, i.e., maximal traces $\langle A_0, c_0 \rangle \to_{\mathcal{T}} \langle A_1, c_1 \rangle \to_{\mathcal{T}} \ldots$ where A_i are agents and $c_i \vdash c_{i-1}$ are constraints. We denote by $(\mathit{Agent} \times \Sigma)^{\vec{\infty}}_\Gamma$ this set of traces. The standard syntax and transition-system semantics is in Table 2. The maximal-trace semantics of a *ccp* program $P = D.A$ is immediately defined as the set of finite and infinite consistent traces of constraints generated from P in an initial store $c \in \Sigma$.

$$
\mathcal{O}(D.A)(c) = \left\{ \delta \in (\mathit{Agent} \times \Sigma)^{\vec{\infty}}_\Gamma \;\middle|\; \begin{array}{l} \delta_0 = \langle A, c \rangle, \forall i \in [0, |\delta|) \\ \delta_i \longrightarrow_{\mathcal{T}} \delta_{i+1} \end{array} \right\}
$$

We define the maximal trace semantics of a *ccp* program P as follows $[\![P]\!]_{\vec{\infty}} = \beta(\{\ \mathcal{O}(P)(c) \mid c \in \Sigma\ \})$, where the

Syntax	Operational semantics
$\begin{array}{lcl} \text{Program} & ::= & \text{Dec . Agent} \\ \text{Dec} & ::= & \epsilon \\ & \mid & p(\vec{x}) \ :- \ \text{Agent . Dec} \\ \text{Agent} & ::= & \mathbf{tell}(c) \\ & \mid & \exists \vec{x}.\text{Agent} \\ & \mid & \text{Agent} \parallel \text{Agent} \\ & \mid & \sum_{i=1}^{n} \ (\mathbf{ask}(c_i) \rightarrow \text{Agent}_i) \\ & \mid & p(\vec{y}) \end{array}$	**R1** $\langle \mathbf{tell}(c)\, ,\, \sigma \rangle \longrightarrow_{\mathcal{T}} \langle \epsilon\, ,\, \sigma \wedge c \rangle$ **R2** $\dfrac{\sigma \vdash c_i}{\langle \sum_{i=1}^{n} \ (\mathbf{ask}(c_i) \rightarrow A_i)\, ,\, \sigma \rangle \longrightarrow_{\mathcal{T}} \langle A_i\, ,\, \sigma \rangle}$ **R3** $\dfrac{\langle A\, ,\, \sigma \rangle \longrightarrow_{\mathcal{T}} \langle A'\, ,\, \sigma' \rangle}{\begin{array}{c}\langle A \Vert B\, ,\, \sigma \rangle \longrightarrow_{\mathcal{T}} \langle A' \Vert B\, ,\, \sigma' \rangle \\ \langle B \Vert A\, ,\, \sigma \rangle \longrightarrow_{\mathcal{T}} \langle B \Vert A'\, ,\, \sigma' \rangle\end{array}}$ **R4** $\dfrac{\langle A\, ,\, d \wedge \exists_{\vec{x}} \sigma \rangle \longrightarrow_{\mathcal{T}} \langle B\, ,\, e \rangle}{\langle \exists(\vec{x}, d).A\, ,\, \sigma \rangle \longrightarrow_{\mathcal{T}} \langle \exists(\vec{x}, e).B\, ,\, \sigma \wedge \exists_{\vec{x}} e \rangle}$ **R5** $\dfrac{p(\vec{x}) :- A \in P}{\langle p(\vec{y})\, ,\, \sigma \rangle \longrightarrow_{\mathcal{T}} \langle \exists(\vec{x}, d_{\vec{x},\vec{y}}).A\, ,\, \sigma \rangle}$

Table 2: The syntax and operational semantics of ***ccp***

function $\beta : \wp((Agent \times \Sigma)^{\vec{\infty}}) \xrightarrow{\text{a}} \wp(\Sigma^{\vec{\infty}})$ abstracts away the agent information from traces:

$$\beta(X) = \left\{ \ \sigma \in \Sigma^{\vec{\infty}} \ \middle| \ \delta \in X, \delta_i = \langle A, \sigma_i \rangle \ \right\}.$$

The following result characterizes both the closure-based denotational [29] and the predicate-transformer semantics of nondeterministic *ccp* programs $P \in Program$ in [13] as abstract interpretations of the maximal trace semantics $[\![P]\!]_{\vec{\infty}}$.

THEOREM 7.2. *Let P be a ccp program.*

- $[\![P]\!]_{\mathcal{D}} \stackrel{def}{=} \lambda X. \ \bigcup \left\{ \ \alpha^{\blacktriangleright}(\alpha^{r}([\![P]\!]_{\vec{\infty}}))(c) \ \middle| \ c \in X \ \right\}$ *is a linear continuous closure operator on Smith's powerdomain $\wp(\Sigma_{\perp})$, with $\perp$ representing divergence, and it corresponds to the denotational semantics in [29].*
- $[\![P]\!]_{gwp} \stackrel{def}{=} \lambda X. \ \ \bigcup \left\{ \ c \ \middle| \ \alpha^{\blacktriangleright}(\alpha^{r}([\![P]\!]_{\vec{\infty}}))(c) \subseteq X \ \right\} = \alpha^{gwp}([\![P]\!]_{\mathcal{D}})$ *is a co-additive function on $\wp(\Sigma_{\perp})$ and its left adjoint function $\lambda X. \ \bigcup \left\{ \ [\![P]\!]_{\mathcal{D}}(c) \ \middle| \ c \in X \ \right\}$ is the strongest-postcondition semantics in [13].*

We close this section by considering examples of programs and their denotational semantics for the different observable behaviors corresponding to the different complementary semantics in Cousot's hierarchy, namely the angelic, demonic, pseudo-demonic, and infinite semantics. These semantics are all abstractions of $[\![\cdot]\!]_{\mathcal{D}}$. Consider the following programs, where $\mathbf{ask}(true) \rightarrow A$ is denoted A and the starting agent is underlined.

$$\begin{array}{rl} P: & p(x) :- \mathbf{tell}(x = 2) + q(x). \\ & q(x) :- p(x).\ \underline{p(x)} \\ Q: & p(x) :- \mathbf{tell}(x = 2).\ \underline{p(x)} \\ U: & p(x) :- \mathbf{tell}(x = 1) + q(x). \\ & q(x) :- p(x).\ \underline{p(x)} \end{array}$$

It is immediate that $[\![P]\!]_{\mathcal{D}^+} = [\![Q]\!]_{\mathcal{D}^+}$ and $[\![P]\!]_{\mathcal{D}^\partial} \neq [\![Q]\!]_{\mathcal{D}^\partial}$, $[\![P]\!]_{\mathcal{D}^\omega} \neq [\![Q]\!]_{\mathcal{D}^\omega}$, $[\![P]\!]_{\mathcal{D}^\varrho} \neq [\![Q]\!]_{\mathcal{D}^\varrho}$. Moreover $[\![P]\!]_{\mathcal{D}^+} \neq [\![U]\!]_{\mathcal{D}^+}$ and $[\![P]\!]_{\mathcal{D}^\partial} = [\![U]\!]_{\mathcal{D}^\partial}$, $[\![P]\!]_{\mathcal{D}^\omega} = [\![U]\!]_{\mathcal{D}^\omega}$, $[\![P]\!]_{\mathcal{D}^\varrho} = [\![U]\!]_{\mathcal{D}^\varrho}$. As for the demonic, pseudo-demonic and infinite semantics are concerned, consider the following programs:

$$\begin{array}{rl} H: & p(x) :- \mathbf{ask}(x = 1) \rightarrow q(x). \\ & q(x) :- q(x).\ \underline{p(x)} \\ K: & p(x) :- \mathbf{ask}(x = 1) \rightarrow q(x) + \\ & \qquad\qquad \mathbf{ask}(x > 1) \rightarrow \mathbf{tell}(y = 2 * x). \\ & q(x) :- q(x).\ \underline{p(x)} \end{array}$$

In this case $[\![H]\!]_{\mathcal{D}^\omega} = [\![K]\!]_{\mathcal{D}^\omega}$ but $[\![H]\!]_{\mathcal{D}^\partial} \neq [\![K]\!]_{\mathcal{D}^\partial}$. Moreover, if we assume that Σ is a finite domain **FD** constraint system (see [24]), and A is a terminating agent that, given $c \in \Sigma$, generates all the $c' \in \Sigma$ such that $c' \vdash c$, then we have:

$$\begin{array}{rl} G: & p(x) :- p(x) + A.\ \underline{p(x)} \\ R: & p(x) :- A.\ \underline{p(x)} \end{array}$$

and $[\![G]\!]_{\mathcal{D}^\varrho} = [\![R]\!]_{\mathcal{D}^\varrho}$ but $[\![G]\!]_{\mathcal{D}^\partial} \neq [\![R]\!]_{\mathcal{D}^\partial}$.

8. CONCLUSION

We have shown that complementary semantics of transition systems in Cousot's hierarchy of semantics can be systematically constructed by domain complementation in abstract interpretation. This provides both a better insight on semantics designed for characterizing complementary observable properties of programs and the possibility to decompose semantics into most abstract factors involving possibly new semantics (e.g. the pseudo-demonic semantics). We think that other operations for domain construction in abstract interpretation can be used for the systematic derivation of semantics in Cousot's hierarchy. In particular, the compositional denotational semantics of a programming language seems to correspond to the reduced power [11, 22] of the non compositional *inevitable termination semantics* in [8].

9. REFERENCES

[1] S. Abramsky and A. Jung. Domain theory. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 3, pages 1–168. Clarendon Press, Oxford, U.K., 1994.

[2] B. Alpern and F. B. Schneider. Recognizing safety and liveness. *Distributed Comp.*, 2:117–126, 1987.

[3] K. R. Apt and G. D. Plotkin. Countable nondeterminism and random assignment. *J. ACM*, 33(4):724–767, 1986.

[4] G. Birkhoff. *Lattice Theory*. AMS Colloquium Publication, 3rd edition. AMS, Providence, RI, 1967.

[5] M. Comini and G. Levi. An algebraic theory of observables. In M. Bruynooghe, editor, *Proc. of the 1994 Internat. Logic Programming Symp. (*ILPS '94*)*, pages 172–186. The MIT Press, Cambridge, Mass., 1994.

[6] A. Cortesi, G. Filé, R. Giacobazzi, C. Palamidessi, and F. Ranzato. Complementation in abstract interpretation. *ACM Trans. Program. Lang. Syst.*, 19(1):7–47, 1997.

[7] P. Cousot. Abstract interpretation. *ACM Comput. Surv.*, 28(2):324–328, 1996.

[8] P. Cousot. Constructive design of a hierarchy of semantics of a transition system by abstract interpretation (Invited Paper). In S. Brookes and M. Mislove, editors, *Proc. of the 13th Internat. Symp. on Mathematical Foundations of Programming Semantics (*MFPS '97*)*, volume 6 of *Electronic Notes in Theoretical Computer Science*. Elsevier, Amsterdam, 1997. URL: http://www.elsevier.nl/locate/entcs/volume6.html, 25 pages, extended version to appear in *Theor. Comput. Sci.*

[9] P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Conference Record of the 4th ACM Symp. on Principles of Programming Languages (*POPL '77*)*, pages 238–252. ACM Press, New York, 1977.

[10] P. Cousot and R. Cousot. Constructive versions of Tarski's fixed point theorems. *Pacific J. Math.*, 82(1):43–57, 1979.

[11] P. Cousot and R. Cousot. Systematic design of program analysis frameworks. In *Conference Record of the 6th ACM Symp. on Principles of Programming Languages (*POPL '79*)*, pages 269–282. ACM Press, New York, 1979.

[12] P. Cousot and R. Cousot. Inductive definitions, semantics and abstract interpretation. In *Conference Record of the 19th ACM Symp. on Principles of Programming Languages (*POPL '92*)*, pages 83–94. ACM Press, New York, 1992.

[13] F. de Boer, M. Gabbrielli, E. Marchiori, and C. Palamidessi. Proving concurrent constraint programs correct. In *Conference Record of the ACM Symp. on Principles of Programming Languages (*POPL '94*)*. ACM Press, New York, 1994.

[14] E. Dijkstra. Guarded commands, nondeterminism and formal derivation of programs. *Comm. of The ACM*, 18(8):453–457, 1975.

[15] P. Dwinger. On the closure operators of a complete lattice. *Indagat. Math.*, 16:560–563, 1954.

[16] G. Filé, R. Giacobazzi, and F. Ranzato. A unifying view of abstract domain design. *ACM Comput. Surv.*, 28(2):333–336, 1996.

[17] G. Filé and F. Ranzato. Complementation of abstract domains made easy. In M. Maher, editor, *Proceedings of the 1996 Joint International Conference and Symposium on Logic Programming (*JICSLP '96*)*, pages 348–362. The MIT Press, Cambridge, Mass., 1996.

[18] R. Giacobazzi. "Optimal" collecting semantics for analysis in a hierarchy of logic program semantics. In C. Puech and R. Reischuk, editors, *Proc. of the 13th Internat. Symp. on Theoretical Aspects of Computer Science (*STACS '96*)*, volume 1046 of *Lecture Notes in Computer Science*, pages 503–514. Springer-Verlag, Berlin, 1996.

[19] R. Giacobazzi, C. Palamidessi, and F. Ranzato. Weak relative pseudo-complements of closure operators. *Algebra Universalis*, 36(3):405–412, 1996.

[20] R. Giacobazzi and F. Ranzato. Complementing logic program semantics. In M. Hanus and M. Rodríguez Artalejo, editors, *Proceedings of the 5th International Conference on Algebraic and Logic Programming (*ALP '96*)*, Lecture Notes in Computer Science, pages 238–253. Springer-Verlag, Berlin, 1996.

[21] R. Giacobazzi and F. Ranzato. Refining and compressing abstract domains. In P. Degano, R. Gorrieri, and A. Marchetti-Spaccamela, editors, *Proc. of the 24th Internat. Colloq. on Automata, Languages and Programming (*ICALP '97*)*, volume 1256 of *Lecture Notes in Computer Science*, pages 771–781. Springer-Verlag, Berlin, 1997.

[22] R. Giacobazzi and F. Ranzato. The reduced relative power operation on abstract domains. *Theor. Comput. Sci*, 216:159–211, 1999.

[23] L. Henkin, J. Monk, and A. Tarski. *Cylindric Algebras. Part I.* North-Holland, Amsterdam, 1971.

[24] P. V. Hentenryck, V. Saraswat, and Y. Deville. Constraint processing in *cc*(FD). In A. Podelski, editor, *Constraint programming: Basics and trends*, volume 910 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1995.

[25] C. Hoare. An axiomatic basis for computer programming. *Comm. of The ACM*, 12(10):576–580, 1969.

[26] J. Morgado. Note on complemented closure operators of complete lattices. *Portug. Math.*, 21(3):135–142, 1962.

[27] G. Plotkin. A structural approach to operational semantics. Technical Report DAIMI FN-19, Comput. Science Dept., Aarhus Univ., Aarhus, Denmark, 1981.

[28] V. Saraswat. *Concurrent Constraint Programming Languages*. The MIT Press, Cambridge, Mass., 1993.

[29] V. Saraswat, V. A. Rinard, and P. Panangaden. Semantic foundations of concurrent constraint programming. In *Conference Record of the 18th ACM Symp. on Principles of Programming Languages (*POPL '91*)*, pages 333–353. ACM Press, New York, 1991.

[30] Z. Shmuely. The structure of Galois connections. *Pacific J. Math.*, 54(2):209–225, 1974.